UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA DE TRANSPORTES E LOGÍSTICA

DAVID LOURENZO DE OLIVEIRA KAILER

SCHEDULING DE ROBÔS GÊMEOS MÓVEIS EM ATIVIDADES DE PICKING E DELIVERY

Joinville
2024

DAVID LOURENZO DE OLIVEIRA KAILER

SCHEDULING DE ROBÔS GÊMEOS MÓVEIS EM ATIVIDADES DE PICKING E DELIVERY

Trabalho apresentado como requisito para obtenção do título de bacharel em Engenharia de Transportes e Logística, no Centro Tecnológico de Joinville, da Universidade Federal de Santa Catarina.

Orientadora: Dra. Silvia Lopes de Sena Taglialenha

Joinville

2024

DAVID LOURENZO DE OLIVEIRA KAILER

SCHEDULING DE ROBÔS GÊMEOS MÓVEIS EM ATIVIDADES DE PICKING E DELIVERY

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de bacharel em Engenharia de Transportes e Logística, no Centro Tecnológico de Joinville, da Universidade Federal de Santa Catarina.

Joinville (SC), 05 de dezembro de 2024.

Banca Examinadora:

_____

Dra. Silvia Lopes de Sena Taglialenha
Orientadora/Presidente

_____

Dr. Ricardo Jose Pfitscher
Membro
Universidade Federal de Santa Catarina

_____

Me. Natan Bissoli
Membro

Dedico este trabalho a Deus, Família e amigos.

AGRADECIMENTOS

Agradeço a Deus, com quem converso todos os dias e que me dá forças para seguir em frente, enfrentar as adversidades e encontrar os caminhos corretos a trilhar.

Agradeço a minha mãe, que sempre me deu todo o apoio para seguir os caminhos que escolhi e por me ajudar a revisar este trabalho, meu pai, que sempre me incentivou a fazer uma faculdade a minha escolha, e toda minha família, por parte de mãe, pai e minha companheira Ana, que me apoiaram e continua apoiando no caminho que decidi seguir.

Agradeço a todos os meus amigos que fiz ao longo de minha vida e mantenho contato até hoje, os que fiz no jardim de infância, ensino fundamental, ensino médio e faculdade, eles não só me apoiam, mas também me incentiva a nunca me satisfazer com o que conquistei no momento, sempre me mostrando que é possível ir além.

Agradeço a todos que conheci no ambiente da UFSC Joinville, aos meus companheiros do PET CTJ o qual me mostraram o início da jornada acadêmica, e que a faculdade não se resume a apenas as aulas da grade do curso.

Agradeço todos os professores ao qual tive o prazer de ter aula, especificamente a professora Silvia que, a partir do projeto de iniciação cientifica que realizamos, me ensinou o valor e os processos da pesquisa acadêmica, orientando-me no desenvolvimento, publicação e apresentação de artigos. Além do suporte completo até meu último momento da entrega deste trabalho de conclusão de curso.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) por ter financiado minha iniciação científica que deu início as pesquisas deste trabalho.

RESUMO

Apresenta-se neste trabalho o Twin-Robot Pallet Assignment and Scheduling Problem (TRPASP), que consiste em determinar a programação de tarefas realizadas por robôs móveis dois robôs móveis posicionados em extremidades opostas ao longo de um trilho em ambientes de picking e delivery. O problema abrange a identificação de quais produtos devem ser coletados por cada robô, bem como a definição da sequência dessas coletas, com o objetivo de minimizar o tempo total necessário para a execução das tarefas. Além de respeitar as posições de coleta, é necessário atender às posições de entrega para formar os pedidos dos clientes. Para resolver instâncias de pequeno porte, propõe-se um modelo de programação linear inteira mista. Por se tratar de um problema NP-difícil (NP-Hard), foi desenvolvido um método heurístico capaz de obter soluções de boa qualidade em tempo computacional reduzido para instâncias maiores. Adicionalmente, são apresentados algoritmos meta-heurísticos baseados em busca local em vizinhanças variáveis (Variable Neighborhood Search – VNS, e Variable Neighborhood Descent – VND) como estratégias de reotimização. Os resultados mostram que as meta-heurísticas propostas melhoram os resultados da heurística em pelo menos 18% para instâncias de pequeno porte e alcançam melhorias de até 74,1% para instâncias de maior complexidade. Observou-se que o método VNS se destacou pelo menor tempo computacional requerido em instâncias de grande porte, apresentando uma redução média de 26% no tempo, em comparação ao VND, para todos os testes realizados.

Palavras-chave: Robôs gêmeos. Agendamento. Designação. Meta-heurística.

ABSTRACT

This study presents the Twin-Robot Pallet Assignment and Scheduling Problem (TRPASP), which involves determining the scheduling of tasks performed by two mobile robots positioned at opposite ends of a rail in picking and delivery environments. The problem includes identifying which products each robot should pick up and defining the sequence of these pickups to minimize the total time required to complete the tasks. In addition to considering the picking positions, it is also necessary to meet the delivery positions to fulfill customer orders. To solve small-scale instances, a mixed-integer linear programming model is proposed. Given the NP-Hard nature of the problem, a heuristic method is developed to obtain high-quality solutions within reduced computational times for larger instances. Additionally, metaheuristic algorithms based on Variable Neighborhood Search (VNS) and Variable Neighborhood Descent (VND) are presented as reoptimization strategies. The results show that the proposed metaheuristics improve the heuristic solutions by at least 18% for small-scale instances and up to 74% for more complex instances. It was observed that the VNS method stood out for its lower computational time in large-scale instances, with an average reduction of 26% compared to VND across all tests performed.

Keywords: Twin robots. Scheduling. Assignment. Metaheuristic.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMR – Autonomous Mobile Robot
CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
FO - Função Objetivo
FJSP - Flexible Job Shop Scheduling Problem
HCE-TRPASP - Heurística Construtiva Especializada para o TRPASP
PFSP - Permutation Flowshop Sequencing Problem
PDP - Pickup and Delivery Problem
PLIM - Programação Linear Inteira Mista
PRISMA - Preferred Reporting Items for Systematic Reviews and Meta-Analyses
TCC - Trabalho de Conclusão de Curso
TRPASP - Twin-Robot Pallet Assignment and Scheduling Problem
TRPP - Twin-Robot Palletising Problem
TRSP - Twin-Robot Scheduling Problem
VNS - Variable Neighborhood Search
VND - Variable Neighborhood Descend

SUMÁRIO

1. INTRODUÇÃO

A eficiência operacional é um fator determinante para a competitividade no setor industrial, demandando o planejamento estratégico e a otimização de recursos. Nesse contexto, tecnologias emergentes como os robôs móveis autônomos (AMRs – Autonomous Mobile Robots) desempenham um papel crescente, oferecendo soluções para otimizar processos de logística interna e movimentação de materiais.

Esses robôs se destacam por sua flexibilidade e adaptabilidade, sendo capazes de executar tarefas de forma autônoma. Essa característica os torna ideais para atividades repetitivas em ambientes dinâmicos. No entanto, a adoção de AMRs sem o devido planejamento estratégico pode gerar desafios significativos, como custos operacionais elevados, gargalos produtivos e problemas de coordenação entre robôs e operadores humanos. Assim, torna-se essencial planejar e programar as atividades de forma criteriosa, buscando maximizar a produtividade e minimizar desperdícios (Chen e Hu, 2021) (Pinedo, 2016).

A eficiência operacional nas empresas depende fortemente do uso responsável de recursos e de um planejamento adequado das atividades. No contexto do planejamento operacional a compreensão de métodos de otimização torna-se essencial para a tomada de decisões eficazes, tanto na alocação dos recursos necessários quanto na execução coordenada das etapas dos processos.

Nos últimos anos, observa-se um aumento significativo no uso de robôs autônomos nas operações industriais. Com o objetivo de combinar a eficiência em larga escala dos sistemas automatizados com a flexibilidade dos processos manuais, muitas indústrias têm adotado novas tecnologias de automação, incluindo AMRs (DANG et al., 2013).

No entanto, a implementação de tais tecnologias, sem um planejamento adequado, pode resultar em prejuízos financeiros e operacionais, especialmente devido ao custo elevado desses sistemas. Assim, uma programação eficiente é fundamental, pois ela permite determinar, por exemplo, a quantidade mínima de robôs necessária para executar todas as tarefas em tempo otimizado, ou ainda estabelecer a sequência ideal de tarefas para minimizar o tempo total de execução, respeitando as restrições técnicas e operacionais.

Diante desse cenário, surge a necessidade de desenvolver modelos de otimização que possam definir a quantidade ideal de robôs, bem como a sequência e

a distribuição das tarefas de modo eficiente. Esses modelos ajudam a responder perguntas críticas como: Qual robô deve fazer qual tarefa? Qual a melhor sequência de execução das tarefas para minimizar o tempo total de operação? Como garantir que as restrições operacionais sejam atendidas?

Para responder a primeira pergunta pode-se fazer um planejamento da utilização de robôs em ambientes de manufatura, desenvolvendo uma solução para o Problema de Designação (Assignment Problem), conforme descrito por Arenales et al. (2007). Este problema, também conhecido como Designação Generalizada, consiste em modelos de otimização que atribuem um conjunto de tarefas (jobs) a um conjunto de agentes com o objetivo de minimizar os custos, sem considerar a ordem das tarefas designadas.

Para responder a segunda pergunta e definir a sequência ideal de execução das tarefas, pode-se considerar o Problema de Agendamento (Scheduling Problem), que considera os jobs já designados e determina a ordem de realização de cada um, visando minimizar, por exemplo, o tempo total de execução (makespan) (PINEDO, 2016).

Este trabalho se propõe a investigar métodos de otimização aplicados ao agendamento e à distribuição de tarefas de coleta e entrega (picking e delivery) com uso de robôs gêmeos em armazéns, uma versão genérica é observada na Figura 1, um problema específico conhecido como Twin-Robot Pallet Assignment and Scheduling Problem (TRPASP). Problemas similares são discutidos em (Berbeglia et al. 2007) e (Thomassom et al., 2018).

Este estudo se baseia na versão adaptada proposta por Kailer e Taglialenha (2023), que apresenta um novo modelo de Programação Linear Inteira Mista (PLIM) e um método de solução que utiliza uma Heurística Construtiva com componentes aleatórios. Para responder a terceira pergunta a heurística testa a factibilidade das soluções por meio de um algoritmo que constrói uma representação geométrica em um plano cartesiano, indicando as posições dos robôs e a execução das tarefas ao longo do tempo e garantido que nenhuma regra foi comprometida.

Esse método, denominado Heurística Construtiva Especializada (HCE) para o TRPASP, será a base para obtenção de soluções iniciais. A partir dessas soluções, serão desenvolvidas e aplicadas meta-heurísticas baseadas em Variable Neighborhood Search (VNS) e Variable Neighborhood Descent (VND), as quais serão abordadas neste trabalho.

Figura 1 - Robôs gêmeos em ambiente industrial.

Fonte: AllGlass (2017 apud Thomasson et al., 2018).

1.1 OBJETIVOS

Para o desenvolvimento deste trabalho, foram propostos os seguintes objetivos gerais e específicos:

1.1.1 Objetivo Geral

Desenvolver e avaliar métodos de otimização para o agendamento e a designação de tarefas de robôs gêmeos autônomos em operações de picking e delivery em armazéns, visando minimizar o tempo total de execução das tarefas (makespan) e o custo de operação, respeitando restrições operacionais e técnicas.

1.1.2 Objetivo Específicos

- Estudar e adaptar o modelo de Programação Linear Inteira Mista (PLIM) proposto por Kailer e Taglialenha (2023) para o Twin-Robot Pallet Assignment and Scheduling Problem (TRPASP), identificando melhorias e restrições aplicáveis ao contexto do trabalho para determinar o ótimo global para instâncias pequenas;
- Identificar as restrições práticas necessárias para a modelagem (tamanho mínimo do trilho, tempo de viagem, tempo de delivery e tempo de picking);

- Implementar uma Heurística Construtiva Especializada (HCE) para o problema TRPASP, com o objetivo de obter soluções iniciais factíveis para o agendamento e designação de tarefas dos robôs em um ambiente de picking e delivery;
- Aplicar e desenvolver meta-heurísticas baseadas em Variable Neighborhood Search (VNS) e Variable Neighborhood Descent (VND) a partir das soluções iniciais obtidas pela HCE, visando aprimorar a eficiência e reduzir o tempo total de execução das tarefas;
- Identificar um conjunto de vizinhanças mínimas necessárias para garantir resultados satisfatórios na aplicação do VNS e do VND;
- Realizar testes de validação e análise de desempenho das soluções obtidas, comparando os resultados com outros métodos e verificando a adequação do modelo proposto em termos de minimização do makespan e dos custos operacionais;
- Analisar o impacto de alteração de parâmetros na eficiência do método proposto;
- Analisar os resultados e discutir as implicações práticas do modelo de otimização desenvolvido para a implementação de robôs móveis autônomos em armazéns, propondo possíveis aplicações e melhorias futuras.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em seis capítulos, incluindo esta introdução.

No Capítulo 2 apresenta-se a metodologia para o desenvolvimento do trabalho, indicando as etapas necessárias para a obtenção, análise e apresentação dos resultados.

No Capítulo 3 é apresentada a fundamentação teórica para os conceitos necessários para o entendimento do TRPASP e síntese da aplicação do método de revisão sistemática PRISMA.

No Capítulo 4 detalha-se os métodos desenvolvidos para resolver o TRPASP.

Os resultados obtidos com a aplicação dos métodos propostos são apresentados e discutidos no Capítulo 5.

Por fim, no Capítulo 6 apresenta-se a conclusão do trabalho, destacando-se as principais contribuições, os objetivos alcançados e possíveis desenvolvimentos futuros.

2. METODOLOGIA

Nesse capítulo será apresentado a metodologia utilizada nesse trabalho, junto dos tipos de pesquisas realizadas.

2.1 CLASSIFICAÇÃO DO TIPO DE PESQUISA

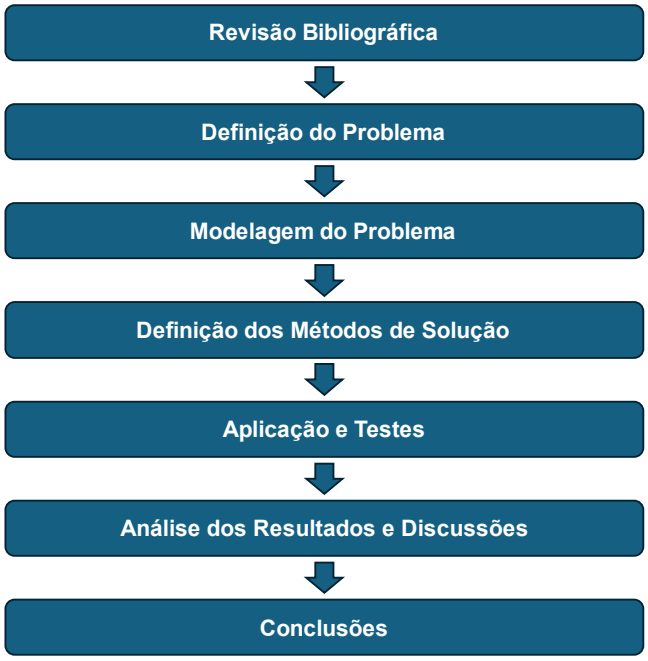
- Natureza da pesquisa: Pesquisa Aplicada, a qual é explicada em (Fleury e Werlang, 2017), foca nos problemas presentes nas atividades de instituições, organizações, grupos ou atores sociais. Essa abordagem dedica-se à elaboração de diagnósticos, identificação de desafios e busca por soluções práticas. Nesse contexto, a pesquisa tem como objetivo desenvolver um modelo aplicável diretamente no ambiente industrial, visando otimizar processos de picking e delivery com o uso de AMRs;
- Abordagem Metodológica: A abordagem utilizada é a Análise Quantitativa dos resultados com base em medidas de desempenho, como o tempo computacional e a função objetivo (FO) gerados pelos modelos de PLIM e heurísticos.
- Objetivo da Pesquisa:
    - Exploratória: Conforme aponta em (Fleury e Werlang, 2017), a pesquisa exploratória busca compreender uma visão geral de um fenômeno ou condição, gerar novas ideias ou identificar os fatos básicos que envolvem uma situação. Neste trabalho, o enfoque exploratório concentra-se na otimização de AMRs, buscando adaptar e propor novos modelos para o contexto do TRPASP.
    - Explicativa: Em (Gil, 2002) é apontado que a pesquisa explicativa tem como foco principal identificar os fatores que determinam ou contribuem para a ocorrência dos fenômenos. Gil (2002) também destaca que esse tipo de investigação aprofunda o conhecimento da realidade, pois busca explicar as razões e os porquês dos acontecimentos. Neste trabalho, a pesquisa explicativa é utilizada para implementar e avaliar o modelo, com o objetivo de compreender os fatores que influenciam as relações entre os parâmetros do problema e as soluções otimizadas, bem como identificar os elementos que contribuem para a eficiência no uso dos AMRs.

- Procedimentos Técnicos: O trabalho segue um Estudo Experimental, explicado em (Gil, 2002) como o processo de determinar um objeto de estudo, selecionar as variáveis que podem influenciá-lo e estabelecer métodos de controle e observação para analisar os efeitos dessas variáveis sobre o objeto. No contexto deste trabalho, isso inclui a realização de testes e validações sobre os modelos e algoritmos apresentados, bem como a utilização de implementações computacionais para avaliar o desempenho do modelo em diferentes cenários.

2.2 ETAPAS DA METODOLOGIA PROPOSTA

Na Figura 2 ilustra-se as etapas da metodologia utilizada para o desenvolvimento do trabalho e na sequência apresenta-se uma breve descrição de cada etapa.

Figura 2 – Etapas de desenvolvimento do trabalho.



Fonte: Autor (2024).

2.2.1 Revisão Bibliográfica

Consiste em um levantamento teórico abrangendo os conceitos de Assignment Problem e Scheduling Problem, com foco em abordagens de otimização aplicadas a ambientes industriais que utilizam AMRs. Essa etapa fundamenta o desenvolvimento do modelo e dos métodos heurísticos empregados, fornecendo

suporte teórico para as estratégias implementadas e sua aplicabilidade no contexto do TRPASP.

2.2.2 Definição do Problema

Esta etapa consiste em compreender o TRPASP, buscando obter uma visão clara do nível de complexidade do problema abordado e como ele se divide em múltiplos problemas clássicos da literatura: Picking e Delivery, Assignment e Scheduling.

2.2.3 Modelagem do Problema

Esta etapa envolve a adaptação e, quando necessário, a reformulação do modelo PLIM para atender às especificidades do TRPASP. Essa etapa contempla a definição detalhada dos parâmetros de cenários específicos para os testes, além da identificação as variáveis, restrições e da função objetivo (FO) do modelo, visando garantir a compatibilidade e a eficiência do modelo no contexto proposto.

Além disso, contempla a análise da estrutura de dados do modelo PLIM e dos modelos heurísticos, com o objetivo de compreender a dimensionalidade e cardinalidade dos parâmetros e variáveis utilizados, bem como a maneira adequada de implementá-los nos softwares necessários.

2.4.4 Definição dos Métodos de Solução

Nesta etapa, utiliza-se o modelo matemático para resolver problemas de menor porte, aumentando gradativamente a complexidade até o ponto em que a resolução em tempo computacional razoável se torna inviável.

Como forma de resolver os problemas maiores é inclui a implementação da HCE-TRPASP para gerar uma solução inicial que forneça uma função objetivo (FO) razoável para o problema. Em seguida, as meta-heurísticas VND e VNS são aplicadas para aprimorar essa solução inicial, buscando alcançar resultados mais próximos do ótimo global.

2.4.5 Aplicação e Testes

São realizados experimentos computacionais utilizando o otimizador GUROBI para execução dos modelos matemáticos. Além disso, é desenvolvido um código em Python para implementar os métodos heurísticos, garantindo flexibilidade no ajuste do tamanho e das características dos cenários estudado.

2.4.6 Análise dos Resultados e Discussões

São analisados os limites do modelo matemático em termos de tamanho das instâncias resolvíveis e, para as instâncias onde o ótimo global é alcançado, avaliada a eficiência dos modelos heurísticos em encontrar essa solução ótima. Para instâncias maiores, resolvidas inicialmente pelo HCE-TRPASP, é examinada a capacidade das meta-heurísticas VNS e VND em melhorar as soluções iniciais. Discute-se, ainda, se há diferenças notáveis no desempenho entre as duas abordagens em termos de qualidade das soluções e tempo computacional.

2.4.7 Conclusões

São interpretados os resultados obtidos, destacando os principais entendimentos gerados ao longo do trabalho. Discutem-se as limitações e as vantagens de cada modelo proposto, considerando sua aplicação em diferentes cenários.

Além disso, são identificadas as contribuições mais relevantes do estudo, suas possíveis implicações práticas para o contexto industrial e sugestões de adaptações para cenários específicos. Por fim, são apresentadas recomendações para trabalhos futuros, com foco em aprimoramentos e novas abordagens que possam expandir o alcance e a eficácia dos modelos desenvolvidos.

No próximo capítulo apresenta-se a fundamentação teórica.

3. FUNDAMENTAÇÃO TEÓRICA

Com o objetivo de compreender os problemas enfrentados na prática, este referencial teórico foi desenvolvido para abordar os conceitos fundamentais relacionados ao tema, bem como apresentar as contribuições de estudos prévios. Por meio da definição geral do problema, da análise dos aspectos abordados na literatura, da exploração dos métodos utilizados para o aprofundamento teórico e da descrição das soluções implementadas, busca-se ampliar o entendimento sobre o TRPASP.

Dessa forma, são investigadas abordagens que possibilitem a resolução eficiente do problema, priorizando soluções viáveis e de qualidade dentro de um tempo computacional razoável.

3.1 DEFINIÇÃO DO PROBLEMA

O TRPASP considerado neste estudo consiste na programação de dois robôs que realizam atividade de coleta e entrega (picking e delivery) em um armazém logístico, como o apresentado na 3, e foi inicialmente estudado em (Thomasson et al., 2022).

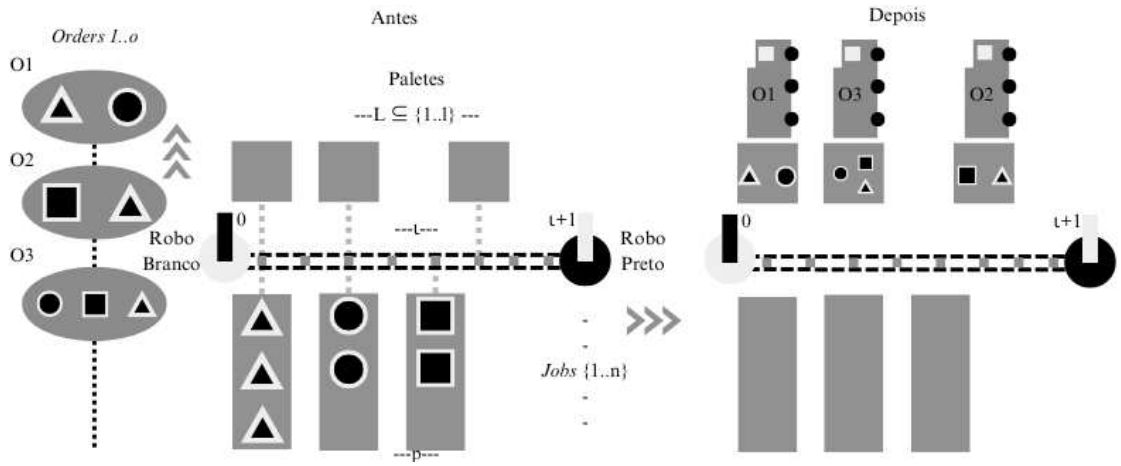
Os robôs (robô branco e robô preto), inicialmente posicionados em lados opostos de um trilho, precisam realizar o picking de ordens de serviço (orders) formadas por conjunto de diferentes produtos (jobs) requisitados por clientes. A quantidade de orders e os tipos de produtos podem variar de acordo com o pedido de cada cliente.

Cada tipo de produto tem um local de coleta (pickup location) previamente definido, e que está posicionado ao longo de um dos lados do trilho, e um local de entrega (delivery location) que está definido em outro ponto no outro lado do trilho. Após realizar a atividade de picking, os robôs devem se movimentar no trilho para realizar o delivery destas orders em posições previamente definidas do outro lado do trilho, e depois retornar às suas posições iniciais.

É considerado um conjunto de orders $(o_1, \ldots, o_m)$, onde cada orders contém um conjunto de jobs $(o_i = \{j_{1i}, j_{2i}, \ldots, j_{ki}\})$, e dado um trilho com l locations, onde é posicionado o robô branco na location 0 e o robô preto na location $l + 1$. A solução do problema consiste em determinar quais jobs serão coletados por qual robô e em quais

delivery locations devem ser entregues, de forma a completar as orders dos clientes. Além disso, é necessário encontrar a melhor sequência de realização dos jobs a fim de minimizar o makespan, ou seja, realizar todas as atividades de picking e delivery.

Figura 3 – Representação do TRPASP.



Fonte: Kailer e Taglialenha (2023).

Para garantir que não haja colisão entre os robôs durante a realização dos jobs, é considerado um parâmetro de distância de segurança, ou seja, uma distância mínima, dada pelas mesmas unidades de locations, que os robôs são permitidos de ficar um do outro. Soluções que não cumpram essa restrição de distância de segurança são consideradas soluções infactíveis.

No cálculo do makespan é levado em consideração o tempo de viagem, que corresponde ao tempo necessário para que o robô se desloque de uma location para outra, e o tempo de processamento, que representa o tempo necessário para realizar uma atividade específica, como o picking de um job em sua pickup location ou o delivery de um job em sua delivery location.

3.2 CONCEITOS NECESSÁRIOS

Nesta seção são abordados os conceitos necessários para o desenvolvimento deste trabalho, tento o objetivo de estabelecer uma visão de termos e ideias das dimensões de problemas dentro do TRPASP. Também é buscado entender onde, dentro da literatura, o TRPASP está localizado.

3.2.1 Picking e Delivery

Pickup and Delivery Problems (PDPs) constituem uma classe de problemas de roteamento de veículos nos quais objetos ou pessoas precisam ser transportados entre origens e destinos, apontam Berbeglia et al. (2007), podendo ter múltiplos contextos, como por exemplo logística de armazéns e depósitos, serviço de transportadora, serviços ambulatoriais, supermercado online, entre outros. Berbeglia et al. (2007) separam os PDPs em três categorias, Many-to-Many Problem, One-to-many-to-one Pickup and Delivery Problem e One-to-one problem.

3.2.1.1 Many-to-Many Problem

Nesse tipo de problema de Picking and Delivery existem múltiplas pickup locations e delivery locations, com a necessidade de encontrar a melhor rota para atender a todas as demandas. Assim, existem múltiplos jobs com múltiplas pickup e delivery locations que precisam ser considerados e desenvolvidos simultaneamente. O objetivo é otimizar a alocação de recursos, como veículos ou funcionários (agentes), para atender a todas as solicitações da maneira mais eficiente possível (BERBEGLIA et al. 2007).

Um dos modelos de many-to-many problem é o Swapping Problem, ilustrado na

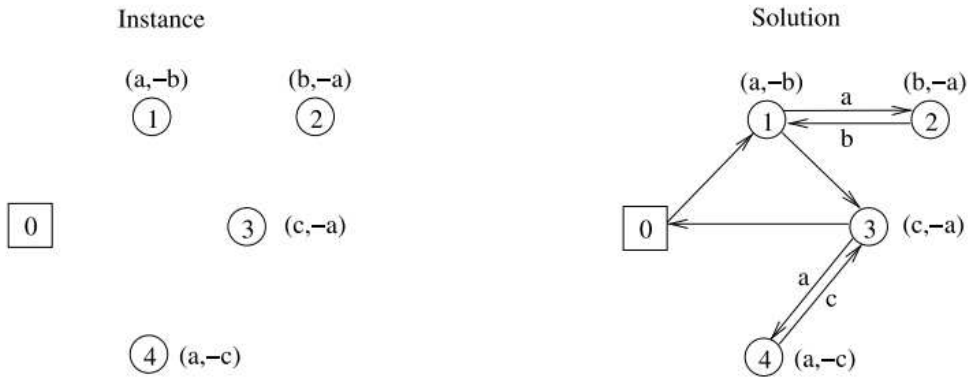
Figura 4, com uma instância à esquerda e uma solução à direita, onde é necessário fazer a reorganização dos itens em cada uma das locations dentro do menor tempo possível (makespan minimizado) ou com a menor quantidade de trocas possíveis (BERBEGLIA et al. 2007).

O Swapping Problem pode ser modelado como um Grafo de n vértices e m arestas, no qual os vértices representam as locations e dentro de cada vértice existem k jobs positivos ou negativos. Se positivo significa que aquela location é a pickup location do job k e, se negativo, representa a delivery location do job k. Por fim as arestas representam a movimentação do agente dentro do meio e é acompanhado de um job k quando a aresta m representa a chegada do k job a sua delivery location (BERBEGLIA et al., 2007).

A dimensão de Picking e Delivery considerada para o TRPASP é entendida como um caso de Many-to-Many Problem, mais especificamente, um caso próximo

ao Swapping Problem, no qual determinadas locations podem atuar tanto como pickup locations quanto como delivery locations. Nesse contexto, os robôs têm como objetivo reorganizar os jobs de forma eficiente para atender todas as orders.

Figura 4 – Representação do Swapping Problem.

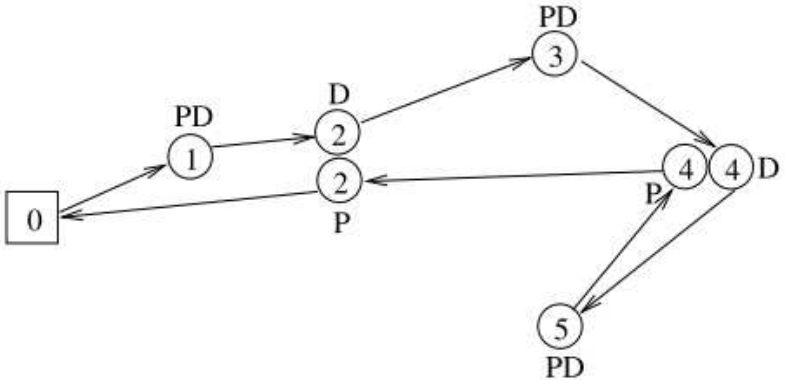


Fonte: Berbeglia et al. (2007).

3.2.1.2 One-to-many-to-one Pickup and Delivery Problem

Berbeglia et al. (2007) representam o One-to-many-to-one Pickup and Delivery Problem (1-M-1-PDP), como um PDP com um único ponto de origem (depósito), múltiplos pontos de pickup, múltiplos pontos de delivery e, em seguida, um ponto de retorno para a origem inicial onde é necessário coletar jobs de diferentes locais (muitos para um), fazer o delivery em diferentes locais e, em seguida, retornar ao depósito.

Figura 5 – Representação do 1-M-1-PDPs com demanda combinatória.



Fonte: Berbeglia et al. (2007).

Um exemplo prático pode ser o caso de um entregador que coleta pacotes de várias lojas e entrega os pacotes em endereços diferentes e, em seguida, retorna ao depósito inicial.

Um exemplo de 1-M-1-PDPs, apontado em (Berbeglia et al., 2007), é sua utilização em demanda combinatória, baseadas em cada job a ser realizado como uma pickup location, indicado por P, uma delivery location, inticado por D, ou ambas, como a ilustrada na

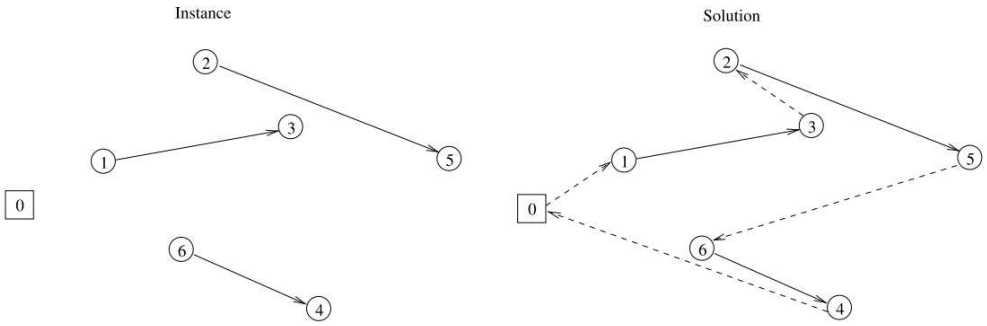
Figura 5.

No contexto do 1-M-1-PDP, a demanda combinatória ocorre quando as pickups e deliveries não são tratadas isoladamente, mas sim como partes interdependentes de uma solução integrada. Assim, a definição e o sequenciamento dos jobs influenciam diretamente a eficiência da rota e o uso dos recursos disponíveis.

3.2.1.3 One-to-one problem

No problema de um a um (One-to-one problem), cada job possui exatamente uma pickup location e uma delivery location. Frederickson et al. (1978 apud Berbeglia et al., 2007) apresentam o Problema da Empilhadeira (Stacker Crane Problem), que envolve o transporte de objetos individuais de sua pickup location até sua delivery location, usando um veículo de capacidade unitária, como ilustrado na Figura 6 com uma instância à esquerda e uma solução à direita.

Figura 6 – Representação do Stacker Crane Problem.



Fonte: Berbeglia et al. (2007).

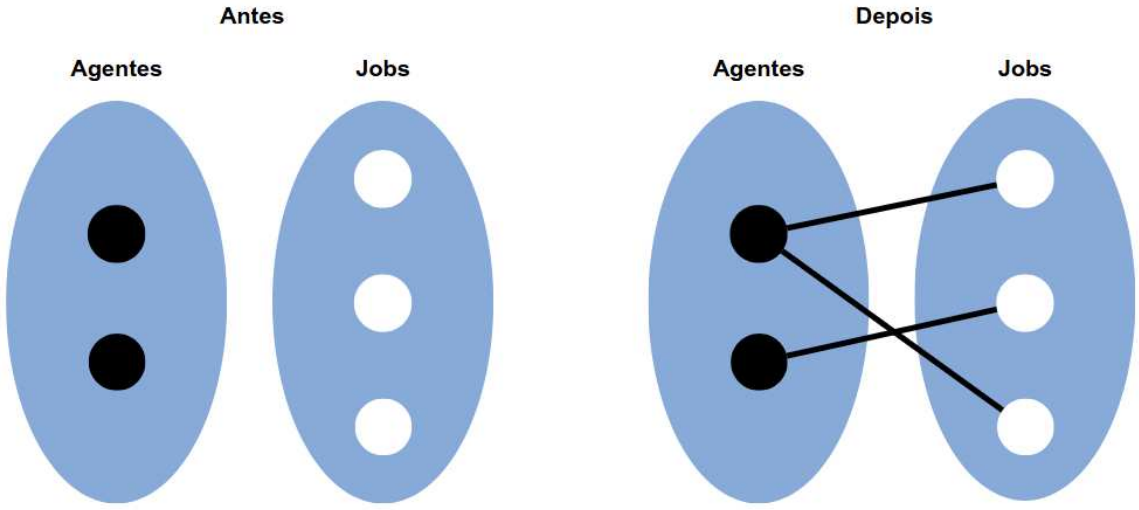
3.2.2 Assignment

O Assignment é a designação entre jobs e agentes, no contexto de múltiplos problemas específicos ilustradas em (Arenales et al., 2007), os que serão vistos neste trabalho são o Assignment Simples e Assignment Generalizado.

3.2.2.1 Assignment Simples

Assignment Simples é visto como o modelo mais simplificado de Assignment, onde se encontra conjuntos de n jobs e m agentes, onde cada agente i é designado a um job j com um custo de manuseio C_{ij}. Em geral, o objetivo do sistema é minimizar a soma de todos os custos C_{ij}. Na Figura 7 é apresentado o problema de Assignment Simples (ARENALES et al. 2007).

Figura 7 – Representação do problema de Assignment Simples.



Fonte: Autor (2024).

Arenales et al. (2007) destacam que este problema ocorre em diversas ocasiões de forma isolada ou como subproblema de situações com problemas mais complexos, exemplos são: designar frações de um lote de um item para processamento de máquinas apontado por LeBlanc et al. (1999 apud Arenales et al., 2007), designar jornais aos centros de impressão dentro de um centro de distribuição apontado por Ree e Yoon (1999 apud Arenales et al., 2007) e designação de clientes

em roteamento de veículos apontado por Fisher e Jaikumar (1981 apud Arenales et al., 2007).

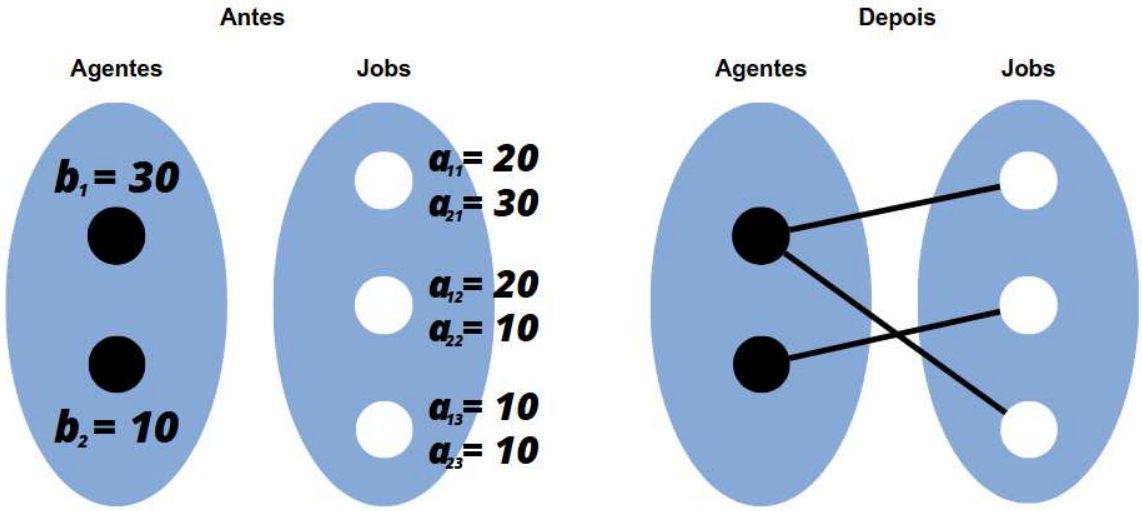
A dimensão de Assignment considerada para o TRPASP é caracterizada como um caso de Assignment Simples, no qual são determinados n jobs e m agentes, sendo $m = 2$ devido à exclusividade do problema para robôs gêmeos. O custo C_{ij} é representado pelo tempo necessário para que o agente i se desloque de sua location até a pickup location do job j em seguida, complete o trajeto até a delivery location correspondente.

3.2.2.2 Assignment Generalizado

Neste caso são considerados dois conjuntos de n jobs e m agentes, com $m < n$. Cada agente i é designado a um job j com um custo de manuseio C_{ij}, onde cada job deve ser realizada por apenas um agente. Junto dessa problemática vem o conceito de capacidade dos agentes, levando que cada agente tem uma capacidade b_i e que ela não deve ser sobrecarregada por uma quantidade a_{ij} de recursos necessários para o job j ser realizada pelo agente i (ARENALES et al., 2007).

O objetivo do problema é minimizar a soma de todos os custos C_{ij}. Na Figura 8 ilustra-se o problema de designação generalizada.

Figura 8 – Representação do problema de Assignment Generalizado.



Fonte: Autor (2024).

3.2.3 Scheduling

Scheduling problem consiste em uma vez determinada a designação dos jobs, também deve-se realizar a ordenação da realização dos jobs. Ou seja, é necessário determinar em que instante de tempo os operadores (ou máquinas) devem iniciar e finalizar a realização de cada tarefa.

Segundo (Taillard. 2023) a operações que precisam ser feitas em uma ordem específica são agrupadas em jobs.

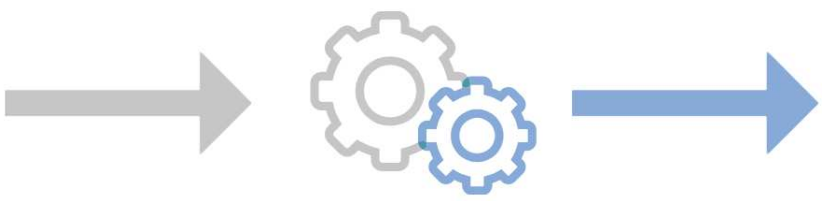
A partir dos múltiplos problemas de Scheduling existentes na literatura, Arenales et al. (2007) destacam os modelos com a programação de máquinas dentro da manufatura, que trazem como principais medidas de desempenho o makespan, tempo de fluxo total, atraso máximo, atraso total e quantidade de jobs atrasados.

Já Pinedo (2016), separa em certas categorias os modelos de Scheduling de máquinas, levando em conta as quantidades de máquinas e os ambientes as quais elas fazem parte, neste trabalho será visto os modelos de uma única máquina, máquinas em paralelo e Job Shop.

3.2.3.1 Modelo de Uma Única Máquina

O ambiente de uma única máquina é um modelo simples, mas um caso especial em vista de todos os outros modelos. Os resultados que podem ser obtidos para modelos de uma única máquina não fornecem apenas compreensão sobre o ambiente de uma única máquina, mas também fornecem uma base para heurísticas que são aplicáveis a ambientes de máquinas de maior complexidade (PINEDO 2016). Na Figura 9 é ilustrada uma representação do modelo de Scheduling com uma única máquina.

Figura 9 – Representação do scheduling de uma máquina.



Fonte: Adaptado de Santo (2014).

Na prática, problemas de Scheduling em ambientes de máquinas mais complexos são frequentemente decompostos em subproblemas que lidam com máquinas individuais (PINEDO 2016).

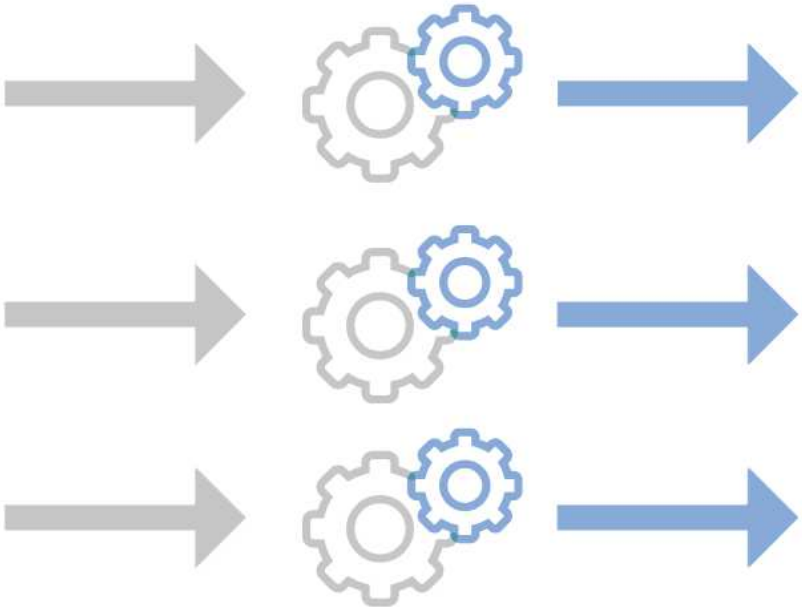
3.2.3.2 Modelo de Máquinas em Paralelo

Pinedo (2016) explica que o modelo de máquinas paralelas trabalha seguindo dois passos principais, primeiramente, designando qual job $i\ \{1..n\}$ será realizada por qual agente $j\ \{1..m\}$ e em seguida determinar a sequência de jobs alocadas para cada máquina.

Na Figura 10 é ilustrada uma representação do modelo de Scheduling com máquinas em paralelo.

O modelo tem três principais objetivos: minimização do makespan, minimização do tempo de conclusão dos jobs e minimização de atraso máximo. Dentro do objetivo de makespan existe uma individualidade, dentro do modelo existe a questão de bom balanceamento de jobs designados às máquinas, mas, quando o makespan é otimizado pelo modelo, a solução assegura que esse balanceamento já será satisfatório (PINEDO 2016).

Figura 10 – Representação do scheduling de máquinas em paralelo.



Fonte: Adaptado de Santo (2014).

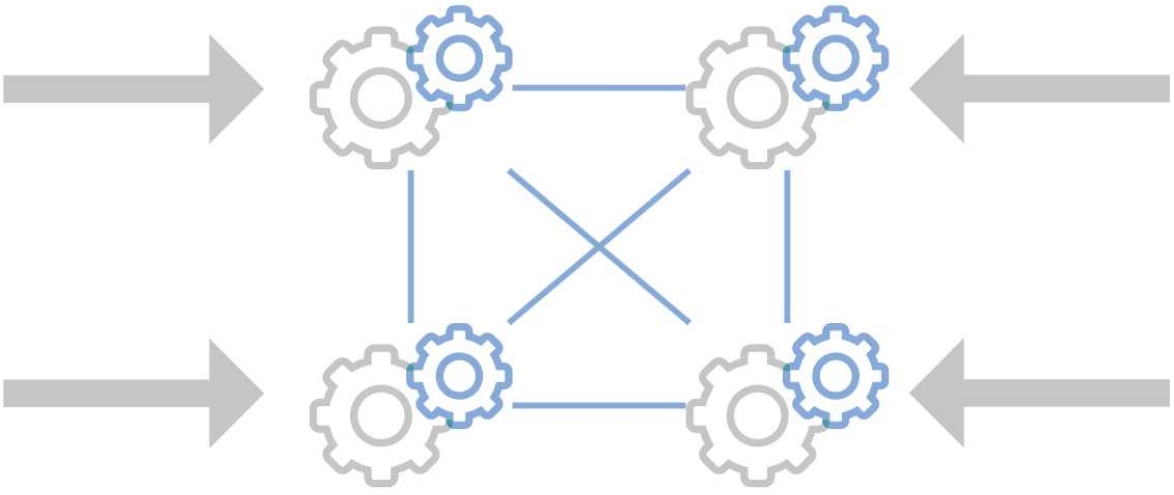
Em (Barbosa et. al, 2021) é apresentado um estudo de caso que considera o scheguling em máquinas paralelas em uma empresa de tubos e conexões para programação de máquinas de injeção plástica.

A dimensão de Scheduling considerada para o TRPASP é caracterizada como um caso de Modelo de Máquinas em Paralelo, onde os robôs gêmeos trabalham em paralelo para realizar todos os jobs necessário, junto da designação de qual robô m, preto ou branco, deverá fazer que job n.

3.2.3.3 Modelo de Job Shop

Seguindo agora para a problemática Job Shop, onde existem múltiplas rotas pré-determinadas para cada job i passando por n agente, sendo possível que um job passe mais de uma vez pela mesma máquina e identificando uma situação de recirculação (PINEDO 2016). Na Figura 11 é ilustrada uma representação do Job Shop.

Figura 11 – Representação do modelo de Job Shop.



Fonte: Adaptado de Santo (2014).

O objetivo mais comum ao se trabalhar em um modelo de Job Shop é na minimização do makespan e as recirculações, mas existem casos que não se consegue evitar o obstáculo da recirculação, exemplo disso é a indústria de semicondutores onde é necessário a recirculação múltipla vezes para completar o processo (PINEDO 2016).

3.3 MÉTODOS DE SOLUÇÃO

Nesta sessão é explorada as formas que podem ser utilizadas para se resolver o TRPASP, onde é considerado tempo computacional, qualidade de solução e técnicas de implementação. As abordagens vistas são: programação linear, problema de programação inteira, problema de programação inteira mista, branch and bound, heurísticas e meta-heurísticas, as quais são detalhadas a seguir.

3.3.1. Métodos Exatos

Os métodos exatos são uma classe de técnicas utilizadas para resolver problemas de otimização, garantindo a obtenção da solução ótima. Como Aguiar, Silva e Mauri (2018) aponta, os métodos exatos calculam a FO de cada solução analisada em busca daquela que oferece o maior valor, em problemas de maximização, ou o menor valor, em problemas de minimização.

3.3.1.1 Programação Linear

Hillier e Lieberman (2010) apontam como modelos lineares são representações matemáticas de problemas de otimização onde tanto a FO quanto as restrições são expressas como equações ou inequações lineares. A programação linear (PL) se busca maximizar ou minimizar uma função linear sujeita a um conjunto de restrições lineares.

3.3.1.2 Programação Linear Inteira

Em muitos problemas práticos, as variáveis de decisão fazem sentido apenas se tiverem valores inteiros. Muitas vezes, é necessário atribuir pessoas, máquinas e veículos a atividades em quantidades inteiras. Se a exigência de valores inteiros for a única maneira pela qual um problema se desvia de uma formulação de programação linear, então trata-se de um problema de programação inteira (IP). O modelo de programação linear inteira (PLI) é o modelo IP com a restrição adicional de que as variáveis devem ter valores inteiros. (HILLIER e LIEBERMAN 2010)

3.3.1.3 Programação Linear Inteira Mista

Como Hillier e Lieberman (2010) dizem, se apenas algumas variáveis são inteiras, enquanto outras podem ser contínuas, ele é conhecido como programação linear inteira mista (PLIM), uma forma mais flexível de PLI. Este modelo é particularmente útil em problemas complexos onde a combinação de decisões discretas e contínuas é necessária. A PLIM permite uma modelagem mais realista de muitos problemas práticos, como aqueles encontrados na engenharia e na economia.

3.3.1.4 Classe NP-hard

Aguiar et al. (2018) apontam que na prática só conseguimos aplicar os métodos exatos em problemas de baixo grau de complexidade, que em geral, está associado ao número de variáveis que fazem parte do problema e, dependendo da complexidade, esses problemas podem ser classificados como NP-Hard.

NP-hard é uma classe atribuída a problemas que não têm soluções conhecidas que possam ser encontradas em tempo polinomial. Isso implica que, conforme a complexidade do problema aumenta, o tempo necessário para o resolver aumenta exponencialmente. Muitos problemas abordados por métodos exatos, como certas versões do problema de Scheduling são NP-hard, como apontado em (Pinedo, 2016). Embora existam métodos capazes de garantir soluções ótimas para esses problemas, sua aplicação em larga escala pode ser inviável devido à elevada demanda computacional.

3.3.1.4 Branch and Bound

O método Branch and Bound é uma técnica de busca sistemática utilizada para resolver problemas com alta complexidade computacional, a fim de retirar análises desnecessárias. Ele funciona com a divisão do problema original em subproblemas, em seguida é relaxado uma ou mais restrições, para estimar rapidamente se um subproblema pode ter uma solução, e resolvido o problema relaxado. A solução ótima do problema relaxado não é necessariamente viável para o problema inicial, mas algumas propriedades interessantes podem ser deduzidas ao resolver o problema relaxado: se o problema relaxado não tiver soluções ou sua

solução ótima for pior do que a melhor solução viável já encontrada, não há necessidade de desenvolver o ramo a partir desse subproblema, como também, se a partir do subproblema encontrado é desenvolvido uma solução ótima viável para o problema inicial, então não é necessário continuar o desenvolvimento do ramo (TAILLARD, 2023).

Técnicas como Branch and Bound são normalmente utilizadas para problemas com alta complexidade computacional como os que são classificados como NP-hard.

3.3.2. Métodos Aproximados

Taillard (2023) aponta que ao tentar resolver novos problemas é utilizado a experiência e conhecimento adquiridos anteriormente, mas dependendo do nível de dificuldade para chegar em uma resolução, é possível aceitar uma solução que não seja o ótimo global. O importante é encontrar uma solução aceitável dentro de um tempo computacional razoável, esse que é o objetivo dos métodos aproximativos.

3.3.2.1 Heurísticas

Para encontrar uma solução aceitável dentro de um tempo computacional razoável as estratégicas heurísticas são utilizadas, onde certas soluções são avaliadas e outras são desprezadas, Aguiar et al. (2018) mencionam quatro técnicas básicas:

- Aleatoriedade: A aleatoriedade permite que a heurística explore o espaço amostral de soluções de maneira casual, sempre em busca de novas soluções sem seguir um caminho específico;
- Gulosinada: A abordagem assume que as combinações que produzem os valores mais altos têm maior probabilidade de resultar em melhores soluções;
- Refinamento: Uma técnica geralmente associada a uma busca local. Quando uma solução é obtida a partir da busca e está próxima da melhor solução do problema, o refinamento, que avalia também os vizinhos de cada solução encontrada, aumenta a probabilidade de encontrar a solução ótima;

- Intensificação: Atua de forma semelhante ao refinamento. Ao identificar uma região promissora no espaço de soluções, a heurística intensifica as buscas nessa área, com a expectativa de encontrar o ótimo global.

3.3.2.2 Meta-heurísticas

Segundo Sorensen, Sevaux e Glover (2018) o termo "meta-heurística" tem sido utilizado para designar duas coisas distintas. A primeira é um framework de alto nível, um conjunto de conceitos e estratégias que se combinam e oferecem uma perspectiva sobre o desenvolvimento de algoritmos de otimização.

Sevaux et al. (2018) também apontam que a segunda acepção do termo "meta-heurística" se refere a uma implementação específica de um algoritmo baseado em tal framework (ou em uma combinação de conceitos de diferentes frameworks) projetado para encontrar uma solução para um problema de otimização específico.

Neste trabalho será utilizadas duas meta-heurísticas com melhor explicação na próxima seção.

3.4 Meta-heurísticas VND e VNS

Em geral algoritmos de busca local em vizinhanças tendem a convergir para ótimos locais, e um dos objetivos das meta-heurística é orquestrar maneiras de se escapar de ótimos locais.

A meta-heurística VND, explicada em (Hansen e Mladenović, 2003), é uma meta-heurística de busca local que, a partir de uma solução inicial, aplica buscas locais sistemáticas em diferentes estruturas de vizinhança, realizando o melhor movimento (busca exaustiva pela solução de melhor FO) na vizinhança da solução de trabalho, e a medida que não existirem melhores soluções na vizinhança atual, expande a busca para vizinhanças maiores, sempre retornando para a vizinhança inicial quando se obtém um sucesso na vizinhança maior. Ou seja, o VND usa a troca sistemáticas de vizinhanças como estratégia para escapar de ótimos locais.

Segundo (Hansen e Mladenović, 2003), dada uma solução inicial S e um conjunto finito de estruturas de vizinhanças $N_S = \{N_1(S), \dots N_{kmax}(S)\}$, o VND segue os passos do Algoritmo 1, em que:

- $K_{\max}$: Quantidade de vizinhanças disponíveis;

- S^*: Representa a melhor solução obtida durante a busca;
- S': Representa a melhor solução alcançada dentro da vizinhança atual;
- f: Representa a função que gera a FO da solução;
- K: Contador de vizinhanças, aumenta quando a melhor função da vizinhança (S') não é melhor que a melhor do sistema (S^*).

Algoritmo 1 Pseudocódigo VND

```
01: Recebe: S, K_max
02:     N_k ← {N_1, N_2, .., N_kmax}
03:     S* ← S
04:     K ← 1
05:     Enquanto K < K_max faça:
06:         S' ← Melhor vizinho ← N_k(S)
07:         Se f(S') < f(S*) então:
08:             S* ← S'
09:             K ← 1
10:         Senão
11:             K ← K + 1
12:         Fim-se
13:     Fim-enquanto
14: Retorna: S*
```

Fonte: Adaptado e traduzido de (Hansen e Mladenovic, 2003).

Algoritmo 2 Pseudocódigo VNS.

```
01: Recebe: S , K_max
02:     N_k ← {N_1, N_2, .., N_kmax}
03:     S* ← S
04:     K ← 1
05:     Enquanto K < K_max faça:
06:         S' ← Turbulência(S, N_k)
07:         S'' ← Melhor vizinho ← N_k(S)
08:         Se f(S'') < f(S*) então:
09:             S* ← S''
10:             K ← 1
11:         Senão
12:             K ← K + 1
13:         Fim-se
14:     Fim-enquanto
15: Retorna: S*
```

Fonte: Adaptado e traduzido de (Mladenovic e Hansen, 1997).

Quando a busca local da linha 6 do Algoritmo 1 também considera movimento aleatórios, tem-se a Variable Neighborhood Search (VNS) descrita no Algoritmo 2. Proposto por Mladenović e Hansen (1997), antes de iniciar a busca local na vizinhança

da solução de trabalho S, realiza-se um movimento aleatório na vizinhança $N_k(S)$ para uma solução S' e, a partir de $N_k(S')$, aplicasse a busca de descida para se obter a solução S''.

Observe que, tanto no VND quanto no VNS, a solução de trabalho é a solução final, pois essas meta-heurísticas só permitem realização de movimentos para soluções de melhor qualidade. A seguir apresenta-se alguns exemplos de aplicação dos métodos VNS e VND.

Focado em Scheduling, o trabalho de Gao, Sun e Gen (2008), apresenta um Algoritmo Genético para resolver o Flexible Job Shop Scheduling Problem (FJSP). Nesse estudo, certos indivíduos do Algoritmo Genético são previamente melhorados por meio de um VND que utiliza duas buscas locais: uma altera um operador, enquanto a outra altera dois operadores.

Zhao et al. (2024) desenvolvem um Simulated Annealing (SA) para um problema de cross-docking. Dentro do framework do SA, a solução é refinada com um VND de cinco estratégias, incluindo trocas determinísticas e aleatórias.

Em (Hansen e Mladenović, 2001) os autores realizam um estudo sistemático do VNS, discutindo sua aplicação na otimização combinatória. O estudo detalha como o VNS alterna entre diferentes estruturas de vizinhança para encontrar soluções ótimas e evitar mínimos locais, destacando sua adaptabilidade e robustez em vários contextos.

Tasgetiren et al. (2007) abordam o Permutation Flowshop Sequencing Problem (PFSP) utilizando um algoritmo de Particle Swarm Optimization integrado ao VNS. Testes em benchmarks amplamente usados mostram melhorias em soluções conhecidas, reduzindo tanto o makespan quanto o tempo total de fluxo em várias instâncias.

Em (Liu et al., 2024) é proposto um modelo para o Flexible Job Shop Scheduling Problem (FJSP), onde o VNS é utilizado de forma auto adaptativa. Integrado a um algoritmo genético, o VNS corrige problemas de viabilidade nas combinações de máquina e dispositivo de fixação, ajustando a solução por meio de iterações que exploram diferentes vizinhanças para minimizar o makespan.

3.5 REVISÃO SISTEMÁTICA DA LITERATURA PARA O TRPASP

Como metodologia de revisão bibliográfica sobre o TRPASP, aplicou-se a metodologia de revisão bibliográfica Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) (Page et al., 2021).

A metodologia PRISMA, segundo Page et al. (2021), é uma ferramenta para realizar coletas de informações sistemáticas em material bibliográfico. Ao seguir as diretrizes do PRISMA, foi possível identificar problemas de Scheduling e Assignment e os principais métodos de solução atualmente empregados.

A metodologia PRISMA adota um protocolo que conduz uma busca sistemática de artigos, dessa forma realiza-se uma seleção de trabalhos a partir dos termos informados e extrai-se dados de maneira padronizada, de forma que se possa avaliar criticamente a qualidade dos estudos a partir dos critérios do pesquisador. Portanto, os artigos incorporados nesta revisão sistemática atendem aos critérios de inclusão e exclusão estabelecidos no protocolo de pesquisa.

A metodologia foi conduzida em quatro fases sequenciais, com o objetivo de identificar pesquisas que abordam o uso dos conceitos de interesse, e são sintetizadas a seguir.

3.5.1 Etapas de Aplicação da Metodologia PRISMA

A seguir são indicadas as etapas desenvolvidas dentro da metodologia PRIMAS.

3.5.1.1 Planejamento

A fundamentação para a elaboração das pesquisas que estruturam este artigo baseou-se na busca por uma base de dados confiável e com um volume significativo de publicações, facilitando a pesquisa de referências e a seleção de documentos relevantes. Isso foi essencial para o desenvolvimento do estudo sobre ferramentas de autoria e sua acessibilidade para diferentes tipos de usuários.

A base de dados selecionada para a investigação foi o portal de periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), que

disponibiliza acesso a uma vasta coleção de periódicos científicos, livros, teses, e outros materiais acadêmicos.

3.5.1.2 Escopo

Este trabalho tem como foco a aplicação de métodos heurísticos de otimização no Scheduling Problem, com ênfase na programação de robôs em ambientes de manufatura. O objetivo principal é explorar a eficiência desses métodos para otimizar o planejamento e alocação de tarefas, considerando o TRPASP. Para tanto, foram investigados também métodos de solução aplicados ao Assignment Problem. A pesquisa abrange o período de 2012 a 2024, com levantamento realizado em maio de 2023 e atualizada em novembro de 2024.

3.5.1.3 Pesquisa

Com base no escopo definido no tópico anterior, foram iniciadas pesquisas aprofundadas para fundamentar os objetivos propostos. Para isso, foi estabelecido um conjunto de palavras-chave que abrangem a temática desta revisão sistemática da literatura, aplicadas à base de dados selecionada.

Utilizando uma metodologia baseada em palavras-chave específicas. As palavras-chave escolhidas para guiar a investigação foram: Robot, Scheduling, Assignment, Makespan e Heuristics. Esses termos foram selecionados por estarem diretamente relacionados aos desafios de programação de robôs e à otimização do makespan (tempo total necessário para a conclusão de um conjunto de tarefas) nos problemas de escalonamento e alocação de tarefas.

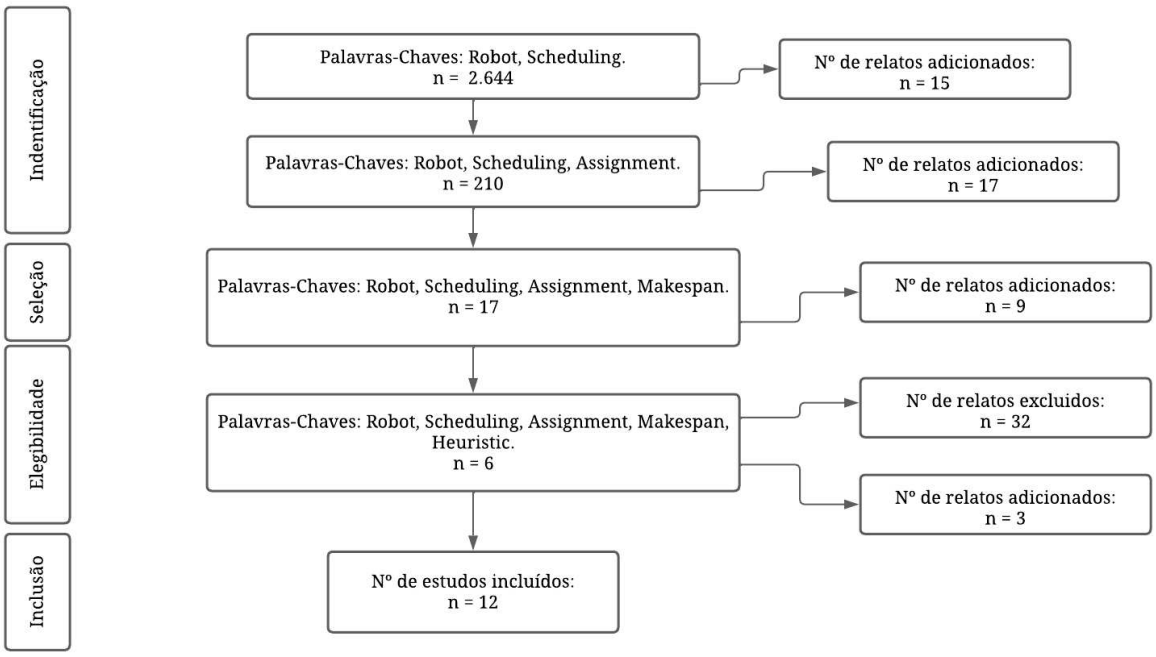
3.5.1.4 Avaliação

Apesar da ampla disponibilidade de documentos resultantes da pesquisa, foi realizada uma filtragem em cada etapa do processo, considerando as adições de palavras-chave. Inicialmente, os artigos foram selecionados com base na avaliação de resumos e palavras-chave. Após a leitura dos trabalhos adicionados, foram eliminados aqueles que apresentavam menor relevância para a proposta do estudo.

Na Figura 12 é apresentado um resumo dos resultados obtidos seguindo a metodologia PRISMA.

Ao final da pesquisa com todas as palavras-chave, foram adicionados 44 artigos. Após a avaliação, 32 foram removidos, totalizando 12 artigos incluídos.

Figura 12 – Fluxograma PRISMA.



Fonte: Autor (2024).

3.5.1.5 Síntese e Análise

Com o objetivo de selecionar as referências a serem incluídas no trabalho, foi realizada uma análise criteriosa dos resultados. Dessa forma, os seguintes trabalhos foram selecionados.

Conforme mencionado anteriormente, o TRPASP foi inicialmente proposto por Thomasson et al. (2022). Os autores desenvolveram um modelo PLIM e apresentaram diversos algoritmos, eles sendo, Variable Neighbourhood Search (VNS), Algoritmo Genético, Algoritmo Húngaro e um Algoritmo Híbrido de alocação de paletes. Com base nos resultados apresentados, verificou-se que o Algoritmo Híbrido obteve os melhores resultados.

A principal base para este TCC foi a pesquisa Kailer e Taglialenha (2023), onde é apresentado um modelo de PLIM modificado de Thomasson et al. (2022), para

não gerar soluções infactíveis e o desenvolvimento de uma heurística chamada Heurística Construtiva Especializada para o TRPASP (HCE-TRPASP) para criar soluções iniciais com componentes aleatórias junto de uma representação geométrica.

Além disso, dois outros trabalhos foram de grande importância para esta pesquisa: Thomasson et al. (2018) e Erdogan, Battarra e Laporte (2014). No trabalho de 2018, o foco foi no Problema de Paletização de Robôs Gêmeos, conhecido como Twin-Robot Palletising Problem (TRPP), o qual apresenta algumas diferenças em relação ao TRPASP. No TRPP, as orders são designadas previamente aos paletes. Em Thomasson et al. (2018) são propostos duas formulações matemáticas e diversos métodos heurísticos para resolver o TRPP. Concluiu-se que, para os dados considerados, o algoritmo genético híbrido foi o mais eficiente na busca de uma solução aceitável em um tempo razoável.

O TRPP é uma versão mais abrangente do Twin-Robot Scheduling Problem (TRSP), que foi apresentado anteriormente em (Erdogan et al., 2014). O TRSP, que pode ser traduzido como Problema de Scheduling de Robôs Gêmeos, envolve a programação dos robôs para realizar os jobs de manipulação de itens, sem considerar a location de pickup dos jobs, uma vez que todos os itens são introduzidos no sistema diretamente no depósito. Para resolver esse problema, foram utilizados dois modelos de Programação Linear Inteira Mista e duas heurísticas construídas especificamente para essa finalidade.

A utilização de soluções heurísticas com robôs autônomos tem se expandido em diversos setores e um dos aspectos mais estudados é o planejamento de trajetórias (Path Planning). Pattnaik, Mishra e Panda (2022), Kanagaraj, Sheik Masthan e Yu (2022), e Sahu, Das e Kabat (2022) exploraram abordagens híbridas e comparativas para encontrar os melhores caminhos em suas respectivas situações.

Wang, Liu e Gombolay (2022) apresenta uma abordagem de scheduleNet para resolver o problema de escalonamento de 5 robôs com testes para múltiplos jobs, levando em consideração jobs heterogêneos. Os resultados obtidos indicam eficiência computacional e um modelo mais genérico para lidar com esses tipos de casos.

Dang, Nguyen e Rudová (2019) apontam uma solução aprimorada para o problema de escalonamento em manufatura flexível, utilizando uma abordagem meta-heurística híbrida que combina Algoritmo Genético e Tabu Search. Esta pesquisa representa uma evolução do trabalho anterior em (Dang et al., 2014).

Fernandes e Taglialenha (2021) apresenta uma nova abordagem para o trabalho anterior de Dang et al. (2019), com uma melhoria no modelo matemático utilizado. Como resultado foi possível obter uma redução de 20% no tempo total de deslocamento do robô no mesmo exemplo considerado. É importante ressaltar que essa adição foi feita com base nas palavras-chave em português.

Tereshchuk et al. (2021) apresentam uma proposta de schedule em jobs de montagem de aeronaves que envolvem múltiplos robôs manipuladores. A abordagem é baseada na designação de jobs para os robôs, levando em consideração restrições de precedência, incorporando elementos de Assignment. O método proposto combina técnicas de machine learning com uma heurística de scheduling, permitindo a seleção automática das relações de precedência adequadas.

3.5.2 Síntese de métodos de solução considerados para o TRPASP

Os métodos de solução empregados no TRPASP foram estudados e analisados em 12 artigos selecionados por meio da aplicação do PRISMA. Conforme detalhado na Tabela 1, esses artigos abordam uma ampla variedade de técnicas heurísticas.

Evidenciado na Tabela 1, o algoritmo genético e os híbridos foram os modelos aproximados mais amplamente utilizados, ambos representando 42% dos métodos empregados, enquanto 67% do trabalhos também utilizaram modelagem matemática.

Algoritmos híbridos são resultado da combinação de conceitos de outros métodos existentes ou de modificações para adaptá-los a contextos diferentes daqueles para os quais foram originalmente desenvolvidos. Essa abordagem pode envolver a combinação de diferentes técnicas heurísticas ou a integração de outras técnicas de otimização aos modelos.

Os resultados obtidos nas pesquisas de Thomasson et al. (2022) e Thomasson et al. (2018), bem como nos estudos de Path Planning conduzidos por Pattnaik et al. (2022), Kanagaraj et al. (2022) e Sahu et al. (2022), apontaram melhorias promissoras no desempenho computacional ao utilizar métodos híbridos.

A popularidade do algoritmo genético deve a sua capacidade de lidar com uma ampla gama de problemas e a eficiência em lidar com múltiplas restrições e soluções em conjunto. No caso do TRPASP, em que as questões de Scheduling, Assignment e Picking e Delivery precisam ser representadas simultaneamente na

solução, o Algoritmo Genético se mostra uma ferramenta altamente favorável para enfrentar esse desafio.

Tabela 1 - Quantidade de trabalhos por método de solução.

Método Heurístico de Solução	**Artigos Incluídos**
Modelo Matemático	67%
Algoritmo Genético	42%
Algoritmos híbridos	42%
Particle Swarm	25%
Heurística Construtiva Própria	17%
Local search	17%
Gravitational Search	17%
Tabu Search	8%
Variable Neighbourhood Search	8%
Branch-and-Bound	8%
Bat Algorithm	8%
Chemical Reaction	8%
Water drop	8%
Whale Optimization Algorithm	8%
Q-Learning	8%

Fonte: Autor (2024).

O Algoritmo Genético, como visto em (Thomasson et al. 2022), é uma técnica de otimização baseada no conceito de evolução natural. Ele utiliza operadores genéticos, como seleção, recombinação e mutação, para explorar o espaço de soluções em busca de uma solução ótima ou próxima da ótima. Sua capacidade de explorar diferentes combinações de soluções torna-o adequado para lidar com problemas complexos como o TRPASP.

4. MÉTODOS DE SOLUÇÃO PROPOSTOS

Tendo em vista que o TRPASP é classificado como NP-Hard (Thomasson et al., 2022), neste trabalho foram propostos uma aplicação de PLIM para instâncias de pequeno porte, e aplicações de métodos heurísticos e meta-heurísticos para resolver instâncias de grande porte.

4.1 SOLUÇÃO CONSIDERANDO PLIM

A primeira abordagem considerada para resolver o TRPASP neste trabalho é pela aplicação de PLIM descrito nas equações (1)-(19), tendo como base o modelo proposto por Kailer e Taglialenha (2023), que foi adaptado de (Thomasson et al., 2022).

4.1.1 Modelo Matemático TRPASP

A seguir apresenta-se as nomenclaturas, parâmetros e variáveis para a formulação do PLIM, tendo como base o modelo proposto em (Kailer e Taglialenha, 2023) onde a Restrição (6) é ajustada:

- $J = \{1,..,n\}$ representa o conjunto dos jobs*;*
- $O = \{1,..,o\}$ representa o conjunto das orders;
- $L \subseteq \{1,..,l\}$ representa o conjunto das pallets location*s* (onde as orders podem ser designadas);
- o_i representa a order que contém o job $i \in J$;
- p_i representa a pickup location referente ao job $i \in J$;
- η representa o tempo de processamento, tempo necessário para um dos robôs pegar ou deixar um dos produtos;
- σ representa a distância segura, distância que deve ser mantida entre os robôs para não ocorrer acidentes;
- τ representa o tempo de locomoção que o robô leva para se mover de uma location para a location vizinha;
- $\alpha_{ii'kk'}$ é a matriz com os valores de tempo mínimo entre a diferença do começo do job i e i' para a distância de segurança σ ser mantida, os detalhes são que o job i

é realizado pelo Preto e indo para a location k e o job i' é realizado pelo Branco e indo para a location k' com o job i sendo realizado antes de i';

- $\beta_{ii'kk'}$ é a matriz com os valores de tempo mínimo entre a diferença do começo do job i e i' para a distância de segurança σ ser mantida, os detalhes são que o job i é realizado pelo Preto e indo para a location k e o job i' está sendo realizado pelo Branco e indo para a location k' com o job i sendo realizado depois de i'.

As matrizes α e β são obtidas a partir de algoritmos específicos para não afetar a complexidade computacional, conforme indicado no Apêndice A para a matriz α, e no Apêndice B, para a matriz β. Neste trabalho considerou-se $\eta = \sigma = \tau = 1$.

São definidas as seguintes variáveis de decisão:

- $x_{ij} = 1$, se o job i é o j-ésimo job a ser feito pelo Branco, e 0 caso contrário;
- $y_{ij} = 1$, *se o* job i é o j-ésimo job a ser feito pelo Preto, e 0 caso contrário;
- $z_{ok} = 1$, *se* order o é designada ao palete k, e 0 caso contrário;
- t_j é o tempo de início do job organizado para se começar na j-ésima vez;
- e_j é o tempo de término do job organizado para ser iniciado na j-ésima *vez;*
- W é o makespan.

A Equação (1) define a FO para minimizar o makespan. O makespan é calculado nas Restrições (12) e (13). A Restrição (2) garante que toda location que tem um palete deve ter no máximo uma order. A Restrição (3) representa que toda order deve estar ligada a apenas uma pallet locations. As Restrições (4) e (5) garantem que cada job deverá ser realizado por um e apenas um dos robôs e ocupa apenas uma posição na programação.

A Restrição (6) deve assegurar o tempo inicial tanto do primeiro job do robô Branco quanto do Preto, ressaltando novamente que os robôs sempre começam as atividades nas margens do trilho, então o Branco está localizado na posição 0 enquanto o Preto começa na posição (l+1), assim, a partir do momento que o primeiro job é designado, qualquer job subsequente pode ser designado como o primeiro job do robô não responsável pelo primeiro job, desta forma, todos os jobs devem seguir a restrição de tempo do primeiro job, para não causar infactibilidade.

O modelo matemático para o TRPASP proposto neste trabalho é então definido pelas Equações e Inequações (1)-(19):

$$Minimizar\ W \tag{1}$$

Sujeito à:

$$\sum_{o \in O} z_{ok} \leq 1 \qquad k \in L \tag{2}$$

$$\sum_{k \in L} z_{ok} = 1 \qquad o \in O \tag{3}$$

$$\sum_{i \in J} (x_{ij} + y_{ji}) = 1 \qquad j \in J \tag{4}$$

$$\sum_{j \in J} (x_{ij} + y_{ij}) = 1 \qquad i \in J \tag{5}$$

$$t_j \geq \sum_{i \in j} \tau p_i x_{ij} + \sum_{i \in j} \tau\,(l + 1 - p_i)\, y_{ij} \qquad j \in J \tag{6}$$

$$e_j \geq t_j + (2\eta + \tau|p_i - k|)(x_{ij} + y_{ij} + z_{o_i k} - 1) \qquad i, j \in J; k \in L \tag{7}$$

$$t_j \geq e_{j'} + \tau|p_i - k| z_{o_{i'} k} - M(2 - x_{ij} + x_{i'j'}) \qquad i, i', j \in J; 1 \leq j' < j; k \in L \tag{8}$$

$$t_j \geq e_{j'} + \tau|p_i - k| z_{o_{i'} k} - M(2 - y_{ij} + y_{i'j'}) \qquad i, i', j \in J; 1 \leq j' < j; k \in L \tag{9}$$

$$t_{j'} \geq t_j + \beta_{ii'kk'} + M(x_{ij} + y_{i'j'} + z_{o_i k} + z_{o_{i'} k'} - 4) \qquad i, i', j \in J; 1 \leq j < j'; k, k' \in L \tag{10}$$

$$t_j \geq t_{j'} + \alpha_{ii'kk'} + M(x_{ij} + y_{i'j'} + z_{o_i k} + z_{o_{i'} k'} - 4) \qquad i, i', j \in J; 1 \leq j' < j; k, k' \in L \tag{11}$$

$$w \geq e_j + \tau \cdot k \cdot z_{o_i k} - M(1 - x_{ij}) \qquad i, j \in J; k \in L \tag{12}$$

$$w \geq e_j + \tau(l + 1 - k) z_{o_i k} - M(1 - y_{ij}) \qquad i, j \in J; k \in L \tag{13}$$

$$x_{i,j} \in \{0,1\} \qquad i, j \in J \tag{14}$$

$$y_{i,j} \in \{0,1\} \qquad i, j \in J \tag{15}$$

$$z_{o,k} \in \{0,1\} \qquad k \in L\,; o \in O \tag{16}$$

$$t_j \geq 0 \qquad j \in J \tag{17}$$

$$e_j \geq 0 \qquad j \in J \tag{18}$$

$$w \geq 0 \tag{19}$$

A Restrição (7) define os tempos de término de todos os jobs, considerando o tempo inicial, tempo de picking e delivery de cada job, e o tempo de locomoção entre as locations. As Restrições (8) e (9) garantem que o job i é designado a um dos dois robôs, (8) para o Branco e (9) para o Preto, como também garante que o tempo de início do robô respectivo não seja menor que o tempo de término somado com o tempo de viagem p_i do robô respectivo.

A Restrição (10) utiliza a matriz β para garantir que o tempo de início do job do Branco é tarde o suficiente para que a distância segura até o Preto seja mantida.

A Restrição (11) utiliza a matriz α para garantir que o tempo de início do job do Preto é tarde o suficiente para que a distância de segurança com o Branco seja mantida.

As Restrições (12) e (13) garantem que o makespan não seja menor que o tempo de término de nenhum job somado com os tempos de retorno para os depósitos, onde ambos os robôs devem chegar a seus depósitos. As Restrições (14) a (19) explicitam o domínio das variáveis utilizadas.

4.1.2 Implementação do PLIM

Para resolver o modelo de PLIM (1)-(19) foi feita uma implementação (Figura 13) em AMPL (A Mathematical Programming Language), considerada uma linguagem de programação matemática de alto nível (AMPL, 2024), e tendo como compilador o Software AMPLIDE, resolvido pelo Gurobi Optimizer (GUROBI, 2024).

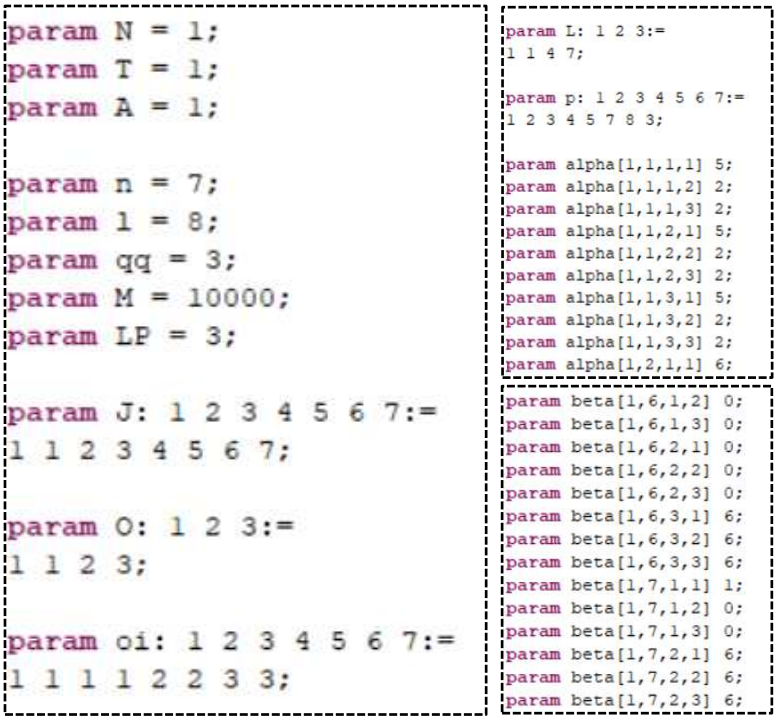
Figura 13 – Implementação em AMPL.

```
#parâmetros
param n;
param qq;
param l;
param LP;
param alpha{i in 1..n, i´ in 1..n,k in 1..LP, k´ in 1..LP};
param beta{i in 1..n, i´ in 1..n,k in 1..LP, k´ in 1..LP};
param J{i in 1..n};
param O{i in 1..qq};
param oi{i in 1..n};
param L{i in 1..LP};
param p{i in 1..n};
param H;
param T;
param A;
param M;
#variáveis
var X{i in 1..n, j in 1..n} binary;
var Y{i in 1..n, j in 1..n} binary;
var Z{o in 1..qq, k in 1..l} binary;
var t{j in 1..n} >= 0;
var e{j in 1..n} >= 0;
var W;
minimize w: W;
subject to C2 {k in 1..LP}: sum{o in 1..qq} Z[o,L[k]] <= 1;
subject to C3 {o in 1..qq}: sum{k in 1..LP} Z[o,L[k]] = 1;
subject to C4 {j in 1..n}: (sum{i in 1..n}(X[i,j] + Y[i,j]) = 1);
subject to C5 {i in 1..n}: (sum{j in 1..n}(X[i,j] + Y[i,j]) = 1);
subject to C6: {j in 1..n}: t[j] >= (sum{i in 1..n} (T*p[i]*X[i,j])) +
               (sum{i in 1..n} (T*(l+1-p[i])*Y[i,j]));
subject to C7{i in 1..n, j in 1..n, k in 1..LP}: e[j] >= t[j] +(2*H  +
               T*abs(p[i]-L[k]))*(X[i,j]+Y[i,j]+Z[oi[i],L[k]]-1);
subject to C8{i in 1..n, i´ in 1..n, j in 1..n, j´ in 1 .. (j-1), k in 1..LP}: t[j] >= e[j´] +
               T*abs(p[i]-L[k])*Z[oi[i´],L[k]] - M*(2-X[i,j]-X[i´,j´]);
subject to C9{i in 1..n, i´ in 1..n, j in 1..n, j´ in 1 .. (j-1), k in 1..LP}: t[j] >= e[j´] +
               T*abs(p[i]-L[k])*Z[oi[i´],L[k]] - M*(2-Y[i,j]-Y[i´,j´]);
subject to C10{i in 1..n, i´ in 1..n, j´ in 1..n, j in 1 .. (j´-1), k in 1..LP, k´ in 1..LP}: t[j´] >=
               t[j] + beta[i,i´,k,k´] + M*(X[i,j]+Y[i´,j´]+Z[oi[i],L[k]]+Z[oi[i´],L[k´]]-4);
subject to C11{i in 1..n, i´ in 1..n, j in 1..n, j´ in 1 .. (j-1), k in 1..LP, k´ in 1..LP}: t[j]   >=
               t[j´] + alpha[i,i´,k,k´] + M*(X[i,j]+Y[i´,j´]+Z[oi[i],L[k]]+Z[oi[i´],L[k´]]-4);
subject to C12{i in 1..n, j in 1..n, k in 1..LP}:e[j] + T*L[k]*Z[oi[i],L[k]] - M*(1-X[i,j]) <= W;
subject to C13{i in 1..n, j in 1..n, k in 1..LP}:e[j] + T*(l+1-L[k])*Z[oi[i],L[k]] - M*(1-Y[i,j]) <= W;
```

Fonte: Autor (2024).

Essa combinação de ferramentas foi empregada para lidar com instâncias de pequeno porte do problema, ou seja, considerando-se instâncias com até $n = 4$ jobs e $O = 1$ order.

Figura 14 – Arquivo de dados de entrada para o PLIM.



```
param N = 1;
param T = 1;
param A = 1;

param n = 7;
param l = 8;
param qq = 3;
param M = 10000;
param LP = 3;

param J: 1 2 3 4 5 6 7:=
1 1 2 3 4 5 6 7;

param O: 1 2 3:=
1 1 2 3;

param oi: 1 2 3 4 5 6 7:=
1 1 1 1 2 2 3 3;
```

```
param L: 1 2 3:=
1 1 4 7;

param p: 1 2 3 4 5 6 7:=
1 2 3 4 5 7 8 3;

param alpha[1,1,1,1] 5;
param alpha[1,1,1,2] 2;
param alpha[1,1,1,3] 2;
param alpha[1,1,2,1] 5;
param alpha[1,1,2,2] 2;
param alpha[1,1,2,3] 2;
param alpha[1,1,3,1] 5;
param alpha[1,1,3,2] 2;
param alpha[1,1,3,3] 2;
param alpha[1,2,1,1] 6;
```

```
param beta[1,6,1,2] 0;
param beta[1,6,1,3] 0;
param beta[1,6,2,1] 0;
param beta[1,6,2,2] 0;
param beta[1,6,2,3] 0;
param beta[1,6,3,1] 6;
param beta[1,6,3,2] 6;
param beta[1,6,3,3] 6;
param beta[1,7,1,1] 1;
param beta[1,7,1,2] 0;
param beta[1,7,1,3] 0;
param beta[1,7,2,1] 6;
param beta[1,7,2,2] 6;
param beta[1,7,2,3] 6;
```

Fonte: Autor (2024).

Para o caso de instâncias com até $n = 7$ e $O = 3$, foi necessário utilizar ao servidor público NEOS-SERVER (AMPL, 2018). Ressalta-se que embora o NEOS-SERVER seja uma opção acessível para processamento de problemas computacionalmente de maior porte, é importante ressaltar que o tempo de processamento nessas instâncias maiores pode ser considerado alto dependendo do ambiente de utilização. Por exemplo, para resolver problemas com até $n = 7$ e $O = 3$, o tempo de processamento foi de aproximadamente 26 minutos, o qual pode ser um tempo baixo para certos contextos e alto para outros.

Os parâmetros qq e LP que aparecem na Figura 13 representam, respectivamente, a quantidade de orders e a quantidade de locations a serem consideradas na instância a ser resolvida.

Na Figura 14 ilustra-se como foi preparado o arquivo de entrada de dados. É importante destacar que apenas uma parte dos parâmetros Alpha e Beta são ilustrados, pois possuem um grande volume de entradas. Ressalta-se ainda que os valores dos parâmetros Alpha e Beta são obtidos por um algoritmo específico apontado em (Thomasson et al., 2022) e ilustrados no Apêndice A e Apêndice B respectivamente.

4.1.3 Classificação do Modelo Matemático TRPASP

Thomasson et al. (2018) verificaram que o TRPASP pode ser reduzido a um problema de Partição, onde é dividindo em conjuntos menores. A verificação é feita a partir uma análise de complexidade, similar à feita por Erdogan et al. (2014) para o TRPP. As análises desenvolvidas na pesquisa comprovam que o TRPASP é um problema NP-Hard, e que instâncias de médio e grande porte não podem ser resolvidas de forma eficiente em tempo hábil. O próprio problema de Partição é conhecido como NP-Completo, o que reforça a complexidade do TRPASP.

4.2 HEURÍSTICA CONSTRUTIVA ESPECIALIZADA PARA O TRPASP

Como PLIM só pode ser aplicado para resolver instâncias pequenas, neste trabalho foi desenvolvida uma heurística denominada heurística especializada para o TRPASP (ou simplesmente, HCE-TRPASP), para determinar soluções factíveis com tempo computacional menor que o do modelo exato e conseguir resolver instâncias de médio e grande porte sem um aumento exponencial de tempo.

A HCE-TRPASP usa uma abordagem de criação de soluções com representação geométrica e matricial para encontrar a melhor solução para o problema em questão. A abordagem geométrica permite uma melhor visualização do problema em geral, enquanto a abordagem matricial permite entender os procedimentos de Picking e Delivery, Scheduling e Assignment que ocorrem na solução.

A HCE-TRPASP constrói soluções iterativamente utilizando componentes aleatórias divididas em 3 vetores, que compõem a matriz de solução, de dimensão n, em que n é o número de jobs do problema, sendo que:

- S_i representa o cronograma de jobs ordenados, sendo assim resolvendo o problema de Scheduling;
- A_i é um vetor booleano que apresenta qual dos robôs realizará cada job, sendo que 0 representa realização pelo robô Branco, e 1 representa a realização pelo robô Preto. Resolvendo, portanto, o problema de Assignment;

- D_i indica a location do destino de cada job, ou seja, indica os locais onde o palete responsável por receber determinada order está localizado do outro lado do trilho, resolvendo assim o problema de Picking e Delivery.

Essa estrutura de dados adotada além de ser de fácil interpretação, também possui intuitividade tanto no manuseio de dados como nas operações do algoritmo.

Na Figura 15 indica-se a representação de uma solução para $n = 7$; $l = 8$; $L = (1,3,6)$; $o = 3$; $o_i = (1,1,2,2,3,3,3)$; $p_i = (1,3,5,1,3,5,1)$, com FO (makespan) dado por $W = 32$.

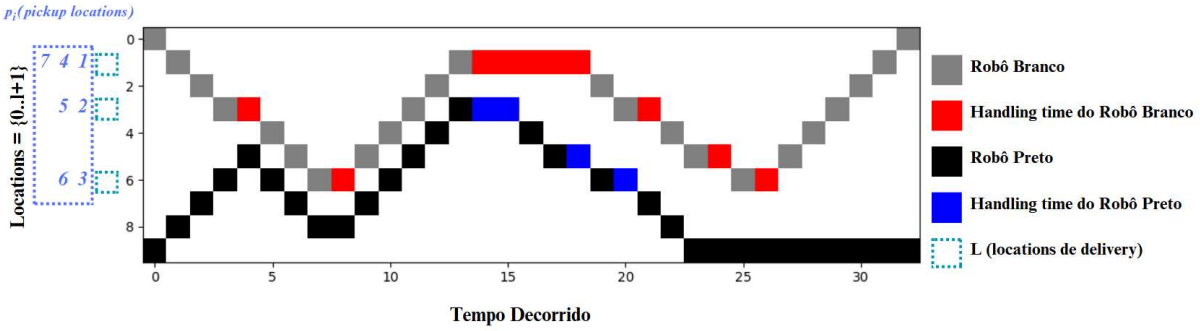
Para validar a factibilidade de cada solução $[S_i, A_i, D_i]$ obtida pela HCE-TRPASP, foi desenvolvido neste trabalho, um procedimento para construir uma representação geométrica da solução analisada, como ilustrado na Figura 16 para a solução dada na Figura 15.

Figura 15 – Estrutura de dados: representação de uma solução.

$$\begin{bmatrix} S_i \\ A_i \\ D_i \end{bmatrix} = \begin{bmatrix} 5 & 2 & 4 & 1 & 7 & 3 & 6 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 6 & 1 & 3 & 1 & 6 & 3 & 6 \end{bmatrix}$$

Fonte: Autor (2024).

Figura 16 – Representação geométrica da solução da Figura 15.



Fonte: Autor (2024).

As informações indicadas no eixo horizontal representam o tempo decorrido ao longo da realização das atividades. As informações do eixo vertical representam a quantidade de locations *(l).* As informações do tempo de processamento são representadas pelas cores vermelha para o robô Branco e azul para o robô Preto.

O período em que o robô está realizando a ação de picking ou de delivery de um job em uma pickup/delivery location respectivamente, é representado com um quadrado de cor correspondente (azul para o preto e vermelho para o branco), tornando a representação mais intuitiva. Destaca-se ainda, que cada quadrado indica a movimentação de uma unidade de distância ao longo do trilho.

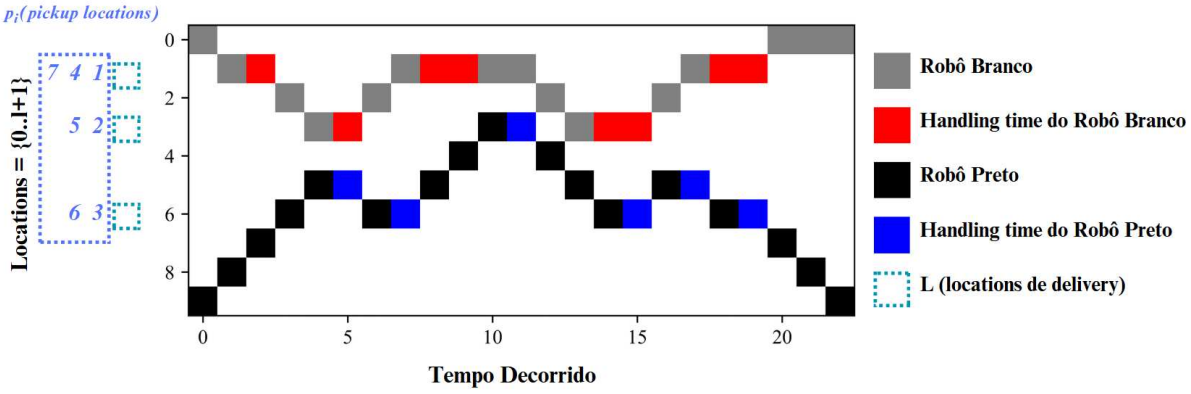
No exemplo ilustrado na Figura 18 tem-se a representação de uma solução ótima para o exemplo considerado, a estrutura de dados da solução é apresentada na Figura 17, com $W^* = 22$

Figura 17 – Estrutura de dados: solução ótima encontrada.

$$\begin{bmatrix} S_i \\ A_i \\ D_i \end{bmatrix} = \begin{bmatrix} 3 & 6 & 2 & 5 & 4 & 7 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 3 & 6 & 1 & 6 & 3 & 6 & 1 \end{bmatrix}$$

Fonte: Autor (2024).

Figura 18 – Representação geométrica da solução da Figura 17.



Fonte: Autor (2024).

Para melhor entender como se dá a construção da solução pela HCE-TRPASP, a seguir apresenta-se o pseudocódigo (Algoritmo 3), cujos parâmetros considerados são definidos por:

- Job $= \{1, .., nR\}$: Lista de jobs j em ordem de inicialização já dividida para cada robô, nR é a quantidade de jobs designada ao robô R;
- Joblocation $= \{1, .., nR\}$: Lista com as pickup locations. j que possuem o job j a ser pego;
- Location: Location que o robô R se encontra (Input para solução Geométrica);

- Wantedlocation: Location que o robô R tem interesse de estar na atual unidade de tempo;
- Goal: Location que está o objetivo do robô R, sendo uma pickup locations ou uma pallet location;
- Return: Ação de um robô R voltar 1 location, no caso do Branco, $\text{location} - 1$, no caso do Preto, $\text{location} + 1$;
- Stay: Ação para um robô R permanecer na location que está;
- Deposit: Location que se refere aos depósitos do robô branco e robô preto;
- NewLocation: Função que retorna a próxima location na próxima unidade de tempo;
- Priority: Definida em (Kailer e Taglialenha, 2023), com o objetivo de solucionar uma situação em que algum dos robôs precisa acessar uma determinada location que violaria a regra de distância de segurança σ. Esta função verifica qual dos robôs está realizando o job de maior prioridade, utilizando a lista S_i como base, assim a função retorna prioridade ao robô Branco, caso o robô Branco esteja desenvolvendo um job com um índice menor em S_i e retorna prioridade ao robô Preto, caso o robô Preto esteja desenvolvendo um job com um índice menor em S_i.

Algoritmo 3 Cálculo da FO.

Recebe: $White, Black, \sigma, \eta, l, \tau$

$W \leftarrow 0$
$Location_{Branco} \leftarrow 0$
$Location_{Preto} \leftarrow l + 1$

Enquanto existem jobs a serem feitos **faça**:
  $W \leftarrow W + \tau$
  $Goal_R \leftarrow Goal(Job_R)$
  $WantedLocation_R \leftarrow WantedLocation(Goal_R, Location_R)$
  $Location_R \leftarrow NewLocation(WantedLocation_R, Priority, \sigma)$
Enquanto algum Robô não está no deposito **faça**:
  $W \leftarrow W + \tau$
  Se $Location_R \neq Deposit$
    $Location_R(Return)$
  Senão:
    $Location_R(Stay)$

Return: W

Fonte: Autor (2024).

Vale destacar aqui, que diferentemente da heurística proposta por Thomassom et al. (2022), a HCE-TRPASP aqui proposta não requer a utilização das matrizes Alpha e Beta, eliminando assim a necessidade de processar múltiplos algoritmos diferentes para a obtenção de solução viável para o problema.

4.3 META-HEURÍSTICAS DE BUSCA EM VIZINHANÇAS PARA O TRPASP

Como explicado na seção 3.4, as meta-heurísticas *Variable Neighborhood Search* (VNS) e *Variable Neighborhood Descend* (VND) (Mladenovic e Hansen, 1997) utilizam a ideia de explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança durante o processo de busca.

Para resolver o TRPASP neste trabalho desenvolveu-se um algoritmo que considerou sete estruturas de vizinhanças, criadas para realizar movimentos de busca que impactassem tanto no problema de scheduling, quanto no problema de assignment. Procurou-se também, estruturar essas vizinhanças para que a busca nas de menor complexidade computacional tivessem prioridade.

A seguir detalha-se como foram definidas essas vizinhanças, sendo θ a representação da complexidade de cada umas das decidas e como elas aumentam em relação a quantidade de jobs n. $\theta(n)$ representa que o tempo computacional irá crescer linearmente com o aumento dos jobs, enquanto $\theta(n^2)$ representa um crescimento quadrático, consequentemente exponencial, em relação a quantidade de jobs.

- Troca de dois jobs consecutivos (Complexidade $\theta(n)$): Trocar dois jobs no vetor S nas posições i e j, em que i varia de $1..n$ e j é $i + 1$;
- Troca de Agentes Binária (Complexidade $\theta(n)$): Troca de robô designado para certo job no vetor A na posição i, em que i varia de $1..n$;
- Troca de dois agentes consecutivos (Complexidade $\theta(n)$): Troca de robôs designados para certo job no vetor A nas posições i e j, em que i varia de $1..n$ e j é $i + 1$;
- Troca de duas delivery locations consecutivas (Complexidade $\theta(n)$): Trocar as delivery locations de duas orders em um job específico no vetor D na posição i e j, incluindo todos os jobs das duas orders correspondentes, se os jobs forem de orders diferentes. Caso contrário, nenhuma alteração é feita. Em que i varia de $1..n$ e j é $i + 1$;

- Troca de dois jobs e agentes consecutivos (Complexidade $\theta(n)$): Troca de dois jobs no vetor S e robôs designados para certo job no vetor A nas posições i e j, em que i varia de $1..n$ e j é $i + 1$;
- Troca de duas delivery locations quaisquer (Complexidade $\theta(n^2)$): Trocar as delivery locations de duas orders em um job específico no vetor D na posição i e j, incluindo todos os jobs das duas orders correspondentes, se os jobs forem de orders diferentes. Caso contrário, nenhuma alteração é feita. Em que i varia de $1..n$ e j é varia de $1..n$;
- Troca de dois jobs e agente quaisquer (Complexidade $\theta(n^2)$): Trocar dois jobs no vetor S nas posições i e j e trocar robôs designados para certo job no vetor A nas posições i e j, em que i varia de $1..n$ e j é varia de $1..n$.

A ordem das vizinhanças apresentada acima reflete, respectivamente, a sequência com a qual as vizinhanças foram executadas nas meta-heurísticas VND e VNS, priorizando as menos complexas no início e ampliando gradualmente o raio de busca à medida que cada vizinhança é explorada.

5. RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados obtidos com a aplicação dos métodos desenvolvidos para resolver o TRPASP que consistem no modelo PLIM, o método heurístico HCE-TRPASP com componentes aleatórias e as meta-heurísticas VND e VNS. As instâncias utilizadas são formuladas com base no tamanho de orders (O) número de jobs (n) e tamanho do trilho a partir da quantidade de locations (L).

Os algoritmos foram executados em um ambiente computacional com as seguintes especificações para julgar tempo computacional: Intel (R) Core (TM) i5-10300H CPU @ 2.50GHz e 24,0 GB de memória RAM.

5.1 RESULTADOS OBTIDOS COM O MODELO PLIM

Como comentado anteriormente, a abordagem de solução do TRPASP com uso de PLIM, possui a vantagem de assegurar a obtenção da solução ótima global para o problema. No entanto, é somente possível resolver instâncias pequenas com esta abordagem.

Tabela 2 - Resolução de instâncias pequenas.

Instâncias	Tamanho			Solução Ótima	Tempo Computacional (s)
	O	n	L		
I1	1	4	8	21	0,15
I2	2	5	8	23	2,08
I3	3	7	8	22	1570,68

Fonte: Autor (2024).

Com este enfoque, o modelo matemático é implementado em instâncias de diferentes tamanhos para entender o limite em que sua aplicação continua viável em relação ao tempo de computação. Para avaliar esta viabilidade, realizaram-se testes em três instâncias de pequeno porte. O objetivo desses testes foi medir o tempo computacional necessário para alcançar a solução ótima global em cada caso. Os detalhes das instâncias testadas estão apresentados na Tabela 2.

Observa-se que, nas duas primeiras instâncias, o tempo computacional é considerado pequeno. No entanto, ao analisar a terceira instância (I3), ocorre um

aumento significativo no tempo de resolução, o que torna o modelo menos eficaz quando lida com instâncias de tamanho equivalente ou maior.

Essa diferença no tempo computacional entre as instâncias indica que, com o aumento dos valores, a aplicação do modelo matemático pode se tornar inviável em termos de eficiência e rapidez para a obtenção da solução. Esse cenário reforça a necessidade de investigar alternativas para tratar instâncias de maior complexidade, sendo esse o ponto em que métodos heurísticos se mostram mais eficazes.

5.2 RESULTADOS OBTIDOS COM HCE-TRPASP, VND E VNS

Foram realizadas análises de resultados ao se considerar os resultados obtidos por HCE-TRPASP, HCE-TRPASP+VND e HCE-TRPASP+VNS. Destacando que para esses últimos dois métodos, HCE-TRPASP foi utilizado apenas para obtenção da solução inicial.

A primeira análise, na Seção 5.2.1, considerou instâncias menores predefinidas (I1 a I3, Tabelas 3 a 7), nas quais é possível encontrar o ótimo global por meio do modelo matemático, permitindo assim a comparação da efetividade do modelo heurístico em alcançar o resultado ótimo.

A segunda análise, também na Seção 5.2.1, é conduzida com instâncias maiores, geradas aleatoriamente (I4 a I9, Tabelas 3 a 7). Para essas instâncias, utilizou-se como solução inicial a melhor solução obtida pela HCE-TRPASP entre r repetições (1, 10, 100, 500 e 1000) para aplicação do VND e a VNS.

Os testes foram feitos seguindo a seguinte ordem:

- Criar de uma solução aleatória factível $[S_i, A_i, D_i]$;
- Identificar o makespan da solução com HCE-TRPASP;
- Repetir o processo r vezes (HCE-TRPASP nas Tabelas 3 a 7);
- Utilizar a solução encontrada com menor makespan como solução inicial para o VNS e VNS;
- Utilizar o VND e VNS para melhorar a solução inicial até não se encontrar mais melhorias nas vizinhanças (HCE-TRPASP+VND e HCE-TRPASP+VNS nas Tabelas 3 a 7).

5.2.1 Resultados e Discussões para 1, 10, 100, 500 e 1000 Repetição

Nas Tabelas 3 a 7 são apresentados os resultados obtidos para HCE-TRPASP, HCE-TRPASP+VND e HCE-TRPASP+VNS, considerando-se as seguintes nomenclaturas:

- TC (s): Tempos computacionais em segundos;
- Melhor Solução Inicial: Melhor solução inicial encontrada, referente a todas as r repetições do modelo (1, 10, 100, 500 e 1000) para a determinada instância;
- nBest: Vezes que o modelo heurístico chegou na melhor Fo conhecida, considerando todas as r repetições (1, 10, 100, 500 e 1000) em porcentagem;
- AvdDev: Desvio padrão entre as r repetições consideradas.
- TC (s): Tempos computacionais em segundos para processamento de todas as vizinhanças, para cada instância;
- Melhor FO: Melhor solução encontrada pelo modelo meta-heurístico referente a melhoria da solução inicial obtida da quantidade de r repetições;
- Melhoria (%): Porcentagem de redução da FO que o modelo meta-heurístico chegou em relação a solução inicial.

5.2.1.1 Resultados e Discussões para 1 Repetição

Para a aplicação da HCE-TRPASP representada na Tabela 3 considerou-se apenas uma repetição do algoritmo. Assim, é possível observar que não requer muito tempo computacional para obtenção de uma solução para todas as instâncias consideradas.

No entanto, ao se considerar a aplicação da VND e VNS, observou-se que essa abordagem não é muito eficiente para se obter soluções iniciais para aplicação de buscas locais. Isso pode, por exemplo, ser observado nas instâncias I1 e I3 pelo VND e I2 e I3 pelo VNS, em que não se alcançou o ótimo global.

À medida que o tamanho das instâncias aumenta, o tempo computacional para os métodos VND e VNS também cresce, devido à complexidade das vizinhanças exploradas. Entretanto, este aumento ocorre em um ritmo significativamente menor em comparação ao modelo exato.

Tabela 3 - Resultados para 1 repetição por instância.

Tamanho				HCE-TRPSP				HCE-TRPSP + VND			HCE-TRPSP + VNS		
Instância	O	n	L	TC (s)	Melhor Solução Inicial	nBest	AvgDev	TC (s)	Melhor FO	Melhoria (%)	TC (s)	Melhor FO	Melhoria (%)
I1	1	4	8	0,000	27	0,00%	-	0,006	22	18,5%	0,01	21	22,2%
I2	2	5	8	0,000	25	0,00%	-	0,01	25	0,0%	0,01	25	0,0%
I3	3	7	8	0,005	38	0,00%	-	0,02	22	42,1%	0,03	26	31,6%
I4	5	20	10	0,003	141	0,00%	-	0,56	75	46,8%	0,82	65	53,9%
I5	5	50	10	0,004	370	0,00%	-	16,02	112	69,7%	11,07	258	30,3%
I6	5	100	10	0,013	708	0,00%	-	133,33	478	32,5%	114,06	459	35,2%
I7	10	20	20	0,002	254	0,00%	-	1,68	92	63,8%	1,85	72	71,7%
I8	10	50	20	0,007	610	0,00%	-	36,44	162	73,4%	33,21	185	69,7%
I9	10	100	20	0,016	1205	0,00%	-	280,72	312	74,1%	257,35	319	73,5%

Fonte: Autor (2024).

Ao analisar o tamanho das Instâncias, observa-se que o número de jobs é um fator de grande relevância no aumento do tempo computacional aos métodos heurísticos e de todos os testes como visto no aumento entre I4 e I6. Outro ponto é o aumento de Locations e Orders, que ao terem seu valor multiplicado por 2, trazem um aumento no tempo de duas a três vezes maior, como comparado nos pares I4-I7, I5-I8 e I6-I9.

Em relação à capacidade de melhora de soluções das meta-heurísticas a partir da busca local, observa-se uma tendência de aumento na melhoria percentual conforme o tamanho das instâncias cresce. O método VND alcança mais de 45% de melhoria em todas as instâncias a partir da I4, atingindo 74,1% na I9. De forma semelhante, o VNS apresenta mais de 50% de melhoria em todas as instâncias a partir da I4, chegando a 73,5% na I9.

5.2.1.2 Resultados e Discussões para 10 Repetições

Na Tabela 4 tem-se os resultados ao se executar a HCE-TRPAS com dez repetições para obter a solução inicial. Neste caso, observou-se também que o tempo computacional requerido para adquirir a solução inicial permanece menor que 1 segundo, chegando ao máximo 0,15 segundos.

No entanto, essa abordagem inicial ainda não garante uma boa seleção de pontos de partida para os métodos de busca local. Também é possível observar que o VND consegue obter a solução ótima nas instâncias menores, em que a solução obtida a partir da I1 já tinha chegado no ótimo pela solução inicial.

Tabela 4 - Resultados para 10 repetições por instância.

Tamanho				HCE-TRPSP					HCE-TRPSP + VND		HCE-TRPSP + VNS		
Instância	O	n	L	TC (s)	Melhor Solução Inicial	nBest	AvgDev	TC (s)	Melhor FO	Melhoria (%)	TC (s)	Melhor FO	Melhoria (%)
I1	1	4	8	0,00	21	10,00%	2,67	0,005	21	0,0%	0,00	21	0,0%
I2	2	5	8	0,00	29	0,00%	3,68	0,01	23	20,7%	0,02	26	10,3%
I3	3	7	8	0,04	26	0,00%	10,72	0,02	22	15,4%	0,01	26	0,0%
I4	5	20	10	0,01	110	0,00%	20,65	0,79	69	37,3%	0,47	78	29,1%
I5	5	50	10	0,04	279	0,00%	43,39	10,60	155	44,4%	8,25	131	53,0%
I6	5	100	10	0,11	495	0,00%	106,03	131,11	233	52,9%	77,36	237	52,1%
I7	10	20	20	0,02	206	0,00%	37,53	2,24	72	65,0%	0,95	115	44,2%
I8	10	50	20	0,06	432	0,00%	80,22	18,30	177	59,0%	18,01	164	62,0%
I9	10	100	20	0,15	824	0,00%	121,04	167,38	340	58,7%	172,45	327	60,3%

Fonte: Autor (2024).

Já o VNS não consegue alcançar a solução ótima, nem na I2 e nem na instância I3.

Observa-se uma tendência de aumento do TC (s) para as meta-heurísticas conforme se considera instâncias maiores, mas ainda assim observasse que o tempo requerido totais são menores que, o tempo requerido quando se considerou apenas uma repetição. Isso pode ocorrer devido a que se considerar soluções de melhor qualidade como ponto de partida, mas também pode estar relacionado às características ou da natureza do problema considerado.

5.2.1.3 Resultados e Discussões para 100 Repetições

A partir da Tabela 5 é notado a aplicação da HCE-TRPASP com 100 repetições do algoritmo, nela o TC (s) chega a pouco mais de um segundo para as instâncias maiores (I6 e I9) mas se mantem menor que um segundo para as outras instancias. Essa abordagem é a primeira a alcançar o ótimo global já na solução inicial para as três instâncias pequenas, tornando o uso do VND e do VNS desnecessário nestes casos (I1, I2 e I3).

Para instâncias maiores, a capacidade de melhoria de soluções das meta-heurísticas mostra uma tendência contínua de aumento. O método VND apresenta mais de 25% de melhoria em todas as instâncias a partir da I4, atingindo 66,1% na I9. Similarmente, o VNS alcança mais de 19% de melhoria em todas as instâncias a partir da I4 e chega a 64,8% na I9. Comparando esta melhoria com os casos anteriores, nota-se um decremento de cerca de 10% no maior caso, isso decorre pela solução inicial estar partindo de um melhor ponto dado a melhor seleção.

Tabela 5 - Resultados para 100 repetições por instância.

Tamanho				HCE-TRPSP				HCE-TRPSP + VND			HCE-TRPSP + VNS		
Instância	O	n	L	TC (s)	Melhor Solução Inicial	nBest	AvgDev	TC (s)	Melhor FO	Melhoria (%)	TC (s)	Melhor FO	Melhoria (%)
I1	1	4	8	0,02	21	11,00%	2,84	0,005	21	0,0%	0,01	21	0,0%
I2	2	5	8	0,03	23	1,00%	4,79	0,01	23	0,0%	0,01	23	0,0%
I3	3	7	8	0,33	22	3,00%	8,02	0,01	22	0,0%	0,01	22	0,0%
I4	5	20	10	0,10	91	0,00%	15,35	0,52	68	25,3%	0,37	73	19,8%
I5	5	50	10	0,38	223	0,00%	37,40	11,70	139	37,7%	19,76	113	49,3%
I6	5	100	10	1,13	504	0,00%	81,04	122,79	210	58,3%	157,86	215	57,3%
I7	10	20	20	0,19	187	0,00%	35,84	1,78	78	58,3%	2,94	66	64,7%
I8	10	50	20	0,63	403	0,00%	62,90	15,99	154	61,8%	12,48	189	53,1%
I9	10	100	20	1,61	892	0,00%	98,54	247,54	302	66,1%	128,48	314	64,8%

Fonte: Autor (2024).

A melhor solução inicial também pode ser o motivo da grande redução no TC (s) das meta-heurísticas, tendo mais de 60 segundos de redução, na I9, na execução do HCE-TRPSP + VND com uma solução 3% melhor e 190 segundos de redução, na I9, na execução do HCE-TRPSP + VNS com uma solução 1,5% melhor.

5.2.1.4 Resultados e Discussões para 500 Repetições

A Tabela 6 apresenta a aplicação da HCE-TRPASP com 500 repetições do algoritmo, onde o tempo computacional permanece baixo, mas passa a ser mais relevante, atingindo um máximo de 7,75 segundos (I9). Esta abordagem consegue alcançar o ótimo global tanto na instância I1 quanto na I3.

Tabela 6 - Resultados para 500 repetições por instância.

Tamanho				HCE-TRPSP				HCE-TRPSP + VND			HCE-TRPSP + VNS		
Instância	O	n	L	TC (s)	Melhor Solução Inicial	nBest	AvgDev	TC (s)	Melhor FO	Melhoria (%)	TC (s)	Melhor FO	Melhoria (%)
I1	1	4	8	0,14	21	5,40%	2,84	0,006	21	0,0%	0,01	21	0,0%
I2	2	5	8	0,17	24	0,00%	4,68	0,01	24	0,0%	0,01	24	0,0%
I3	3	7	8	1,56	22	1,80%	7,66	0,01	22	0,0%	0,01	22	0,0%
I4	5	20	10	0,55	100	0,00%	15,90	0,46	66	34,0%	0,51	66	34,0%
I5	5	50	10	1,94	210	0,00%	37,67	11,01	108	48,6%	6,65	114	45,7%
I6	5	100	10	5,28	492	0,00%	85,25	86,46	240	51,2%	116,98	233	52,6%
I7	10	20	20	1,02	124	0,20%	34,60	1,10	88	29,0%	2,19	71	42,7%
I8	10	50	20	2,90	408	0,00%	60,79	14,85	140	65,7%	16,83	158	61,3%
I9	10	100	20	7,75	787	0,00%	112,47	126,78	316	59,8%	106,22	347	55,9%

Fonte: Autor (2024).

As meta-heurísticas mantêm a tendência de melhoria com o aumento das instâncias, com uma anormalidade registrada: a melhor porcentagem de melhoria ocorre na instância I8, alcançando 65,7% com o VND e 61,3% com o VNS. Este comportamento decorre da natureza específica da solução inicial alcançada, assim como das características do problema gerado.

Ao se observar a I9, é notado que dentro de todos os testes de r repetições feitas (1,10, 100, 500 e 1000), esse é que chega no resultado com menor TC (s). No entanto, ao analisar a qualidade da solução, é notado que o testes de 1 e 100 repetições, mesmo tendo um TC (s) maior, conseguiram chegar em soluções melhores. Chegando na conclusão de que a velocidade observada no teste com 500 repetições decorre da limitação de melhoria pelas meta-heurística, especificamente pelo VNS que teve a pior solução para a I9 dentro de todos os testes.

5.2.1.5 Resultados e Discussões para 1000 Repetições

Para a aplicação da HCE-TRPASP representada na Tabela 7 considerou-se 1000 repetições do algoritmo, o tempo computacional cresce proporcionalmente, alcançando um máximo de 14,92 segundos na instância I9, aproximadamente o dobro do tempo observado com 500 repetições.

Tabela 7 - Resultados para 1000 repetições por instância.

Tamanho				HCE-TRPSP			HCE-TRPSP + VND				HCE-TRPSP + VNS		
Instância	O	n	L	TC (s)	Melhor Solução Inicial	nBest	AvgDev	TC (s)	Melhor FO	Melhoria (%)	TC (s)	Melhor FO	Melhoria (%)
I1	1	4	8	0,20	21	8,90%	2,90	0,005	21	0,0%	0,03	21	0,0%
I2	2	5	8	0,39	23	0,20%	4,78	0,01	23	0,0%	0,01	23	0,0%
I3	3	7	8	3,13	22	2,30%	7,89	0,01	22	0,0%	0,01	22	0,0%
I4	5	20	10	1,06	90	0,10%	16,10	0,85	73	18,9%	0,43	72	20,0%
I5	5	50	10	4,06	205	0,10%	40,09	5,52	105	48,8%	5,86	104	49,3%
I6	5	100	10	10,57	471	0,10%	85,21	79,78	239	49,3%	66,91	234	50,3%
I7	10	20	20	1,96	147	0,00%	34,81	1,25	74	49,7%	0,79	80	45,6%
I8	10	50	20	5,60	339	0,10%	62,22	12,07	160	52,8%	13,38	162	52,2%
I9	10	100	20	14,92	726	0,10%	112,20	192,15	292	59,8%	132,78	290	60,1%

Fonte: Autor (2024).

Com 1000 repetições é possível alcançar o ótimo global em todas as instâncias menores e identificar a melhor solução inicial em quase todas as instâncias quando comparado com os testes anteriores (1, 10, 100 e 500 repetições), exceto na I7, onde a melhor solução inicial foi obtida com 500 repetições. Esse fenômeno ocorre devido à natureza estocástica do processo.

As meta-heurísticas mantêm a evolução das melhorias conforme o aumento das instâncias. Nota-se que, nos testes de repetições anteriores, a solução para a instância I9 apresentou valores superiores a 300, enquanto o teste com 1000 repetições alcançou uma solução de 292 pelo processo do VND e 290 pelo processo

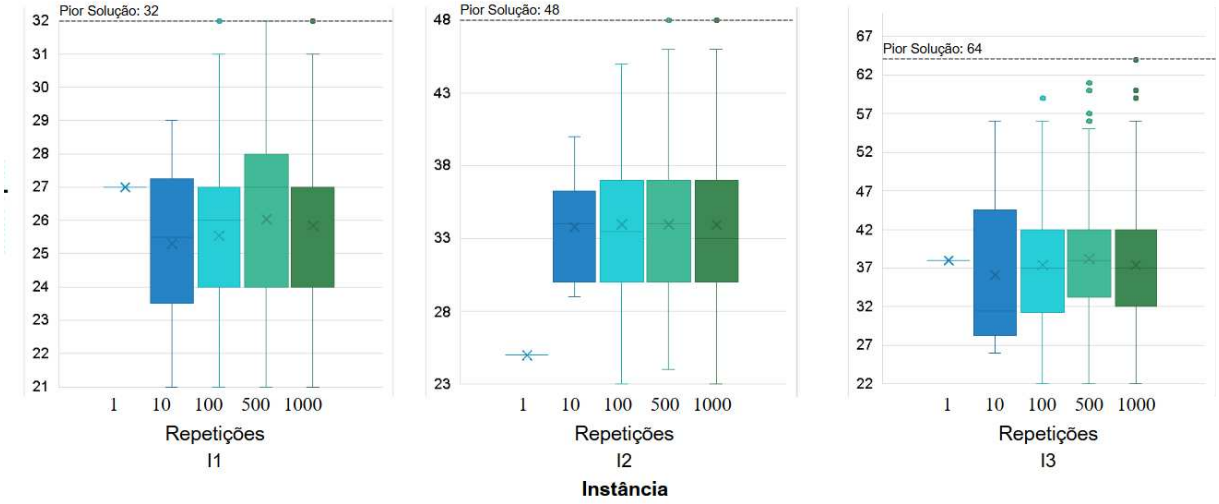
do VNS. Esse resultado confirma a sensibilidade que os modelos de busca local apresentam em relação à qualidade da solução inicial.

5.2.1.6 Resultados e Discussões das Instâncias I1, I2 e I3

Na Figura 19, são apresentados três gráficos com os resultados das instâncias nas quais o modelo matemático encontrou o ótimo global (I1, I2 e I3), representado pelos valores mais baixos em cada gráfico.

Observa-se que, na primeira instância, os testes de 10 a 1000 repetições atingiram a solução ótima; na segunda instância, apenas os testes com 100 e 1000 repetições obtiveram o ótimo; e, na terceira instância, o ótimo foi encontrado com 100, 500 e 1000 repetições.

Figura 19 - Resultados de I1, I2 e I3 por número de repetições.



Fonte: Autor (2024).

Quanto ao AvgDev, as três instâncias mantiveram um padrão semelhante, indicando que, mesmo quando a FO ótima é mais difícil de alcançar, os valores obtidos se mantêm próximos ao ideal.

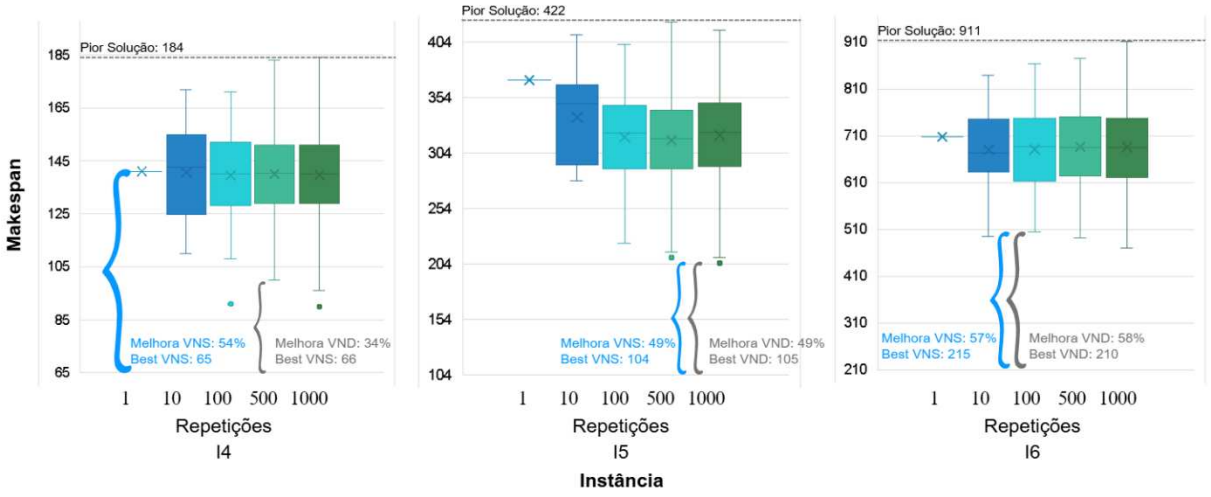
5.2.1.7 Resultados e Discussões das Instâncias I4, I5 e I6

Na Figura 20, considerando instâncias com 5 orders e 10 locations cada (I4, I5 e I6), foi usada a quantidade de jobs de acordo com o estudo de Thomasson et al. (2022) para avaliar a eficiência da heurística e das meta-heurísticas.

Nestas instâncias, percebe-se um aumento acentuado no AvgDev à medida que a complexidade do problema cresce com a adição de mais jobs.

Ao gerar a solução inicial, observa-se a melhor solução encontrada por cada meta-heurística: o VNS alcançou a melhor solução nas instâncias I4 e I5, com melhorias de 54% e 49% nas repetições 1 e 1000, respectivamente; enquanto o VND encontrou a melhor solução na instância I6, com uma melhoria de 58% a partir do teste de 100 repetições do algoritmo.

Figura 20 - Resultados de I4, I5 e I6 por número de repetições.

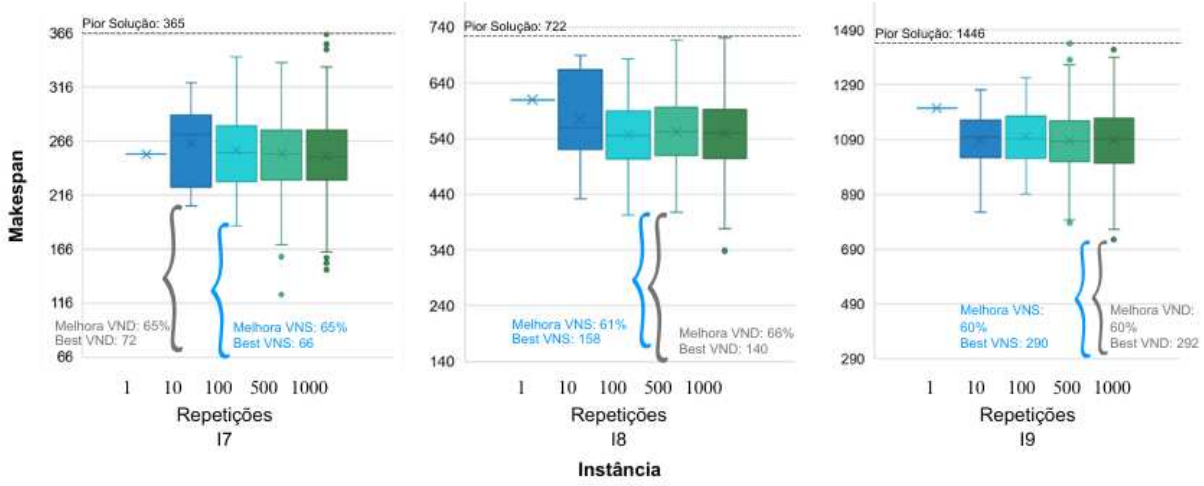


Fonte: Autor (2024).

5.2.1.8 Resultados e Discussões das Instâncias I7, I8 e I9

Para as instâncias ilustradas na Figura 21 (I7, I8 e I9), mesmo com o aumento do número de locations e orders, o TC (s) (Tabela 3 a 7) para criar a solução inicial não apresentou um crescimento expressivo; contudo, o tempo para aplicar melhorias pelas meta-heurísticas aumentou de forma proporcional.

Figura 21 - Resultados de I7, I8 e I9 por número de repetições.



Fonte: Autor (2024).

As melhores solução obtida pelo VNS são alcançadas nas instâncias I7 e I9, com melhorias de 65% e 60% nas repetições 100 e 1000, respectivamente. Já o VND obteve a melhor solução na instância I8, apresentando uma melhoria de 66% na rodada de 500 repetições.

Considerando as instâncias de I4 a I9, nas quais a solução ótima não foi encontrada na etapa inicial (I1, I2 e I3), o VNS obteve a melhor solução em quatro das instâncias (I4, I5, I7 e I9), enquanto o VND obteve a melhor solução nas outras duas (I6 e I8).

Em termos de tempo computacional, o VNS indica ser mais rápido que o VND na maioria das instâncias maiores, quando observado a instancia I9 e todos os testes feitos (1,10, 100, 500 e 1000 repetições), o VNS teve em máximo 48% de redução no tempo computacional e uma média de 26%, evidenciando uma vantagem em velocidade de processamento para problemas de maior complexidade.

6. CONCLUSÃO

Neste trabalho apresentaram-se os resultados do estudo do Twin-Robot Pallet Assignment and Scheduling Problem - TRPASP, no qual dois robôs foram posicionados em lados opostos de um trilho para realizarem operações de picking e delivery. O problema consistiu em definir quais produtos deveriam ser designados para cada robô, indicando o ponto de picking e o ponto de delivery, bem como a sequência de realização das operações de cada robô, com o objetivo de minimizar o tempo total de realização das tarefas e atendendo restrições de segurança para que não ocorresse colisão entre os robôs, sendo classificado como um problema combinatório NP-Hard.

Apresentou-se um modelo de PLIM que determinou a solução ótima global para instâncias de pequeno porte e aplicou-se uma abordagem heurística, denominada HCE-TRPASP, capaz de determinar soluções ótimas para os problemas de menor porte em baixo tempo computacional e soluções viáveis em tempo computacional satisfatório para instâncias maiores.

Também se propuseram algoritmos meta-heurísticos baseados em busca local em vizinhanças variáveis (Variable Neighborhood Search – VNS, e Variable Neighborhood Descent – VND), tendo como solução inicial as soluções obtidas pela HCE-TRPASP. Ambas as meta-heurísticas trouxeram melhorias bastante semelhantes, alcançando pelo menos 18% nas instâncias menores e chegando a 74,1% na maior instância testada. Quanto ao tempo computacional, o VNS indicou uma vantagem sobre o VND, especialmente nas instâncias de maior porte. Dentro de todas as repetições (1, 10, 100, 500 e 1000), quando comparado à maior instância (I9), o VNS teve um máximo de 48% de redução no tempo computacional e uma média de 26%, destacando-se como uma opção mais eficiente para problemas de maior complexidade.

As principais contribuições deste trabalho foram:

- Apresentar uma forma de verificação da factibilidade das soluções obtidas;
- Retirar a necessidade de aplicar múltiplos algoritmos a partir da retirada da necessidade de utilizar as matrizes alpha e beta;
- A HCE-TRPASP para criação de soluções iniciais, com resultados validados para exemplos em que se conhece a solução ótima;

- Análisar das meta-heurísticas VND e VNS para melhoria das soluções iniciais criadas pela HCE-TRPASP.

O estudo realizado no Trabalho de Conclusão de Curso permitiu a aplicação prática dos conhecimentos adquiridos ao longo da graduação e possibilitou o aprofundamento em materiais de relevância acadêmica. Além disso, proporcionou a oportunidade de analisar problemas reais de alta complexidade, promovendo o desenvolvimento de habilidades como pensamento crítico, analítico e criativo. Essa experiência também contribuiu para o fortalecimento de competências como gestão de tempo, organização, resolução de problemas e aplicação de modelos matemáticos, heurísticos e meta-heurísticos, essenciais para a atuação profissional na área.

Para trabalhos futuros recomenda-se:

- Utilizar da HCE-TRPASP aqui proposta para obter soluções iniciais para reotimização com novas meta-heurísticas não avaliadas nesse trabalho;
- Analisar do problema para n robôs.
- Realizar minimização multiobjetivo para reduzir makespan e movimentos dos robôs.
- Aplicar novos movimentos de descida para o VNS e VND;
- Aplicar o modelo heurístico e meta-heurístico propostos a partir de um caso real.

REFERÊNCIAS

AGUIAR, M.; SILVA, R.; MAURI, G. Introdução aos Métodos Heurísticos de Otimização com Python. 2018.

AMPL. **User's Guide to the NEOS Server**. 2018. Disponível em: https://neos-guide.org/content/users-guide. Acesso em nov. 2024.

AMPL. **Why AMPL**. 2019. Disponível em: https://ampl.com/products/ampl/. Acesso em nov. 2024.

ARENALES, M. et al. **Pesquisa operacional**. São Paulo: Elsevier Editora Ltda. 2007.

BARBOSA, L. B.; FERNANDES, C. W. N.; SILVA, V. M. D.; TAGLIALENHA, S. L. S. Modelo para sequenciamento de tarefas em máquinas paralelas heterogêneas de injeção plástica com prazo de entrega. In: ENEGEP 2021 Encontro Nacional de Engenharia de Produção, 2021, Anais. 2021.

BERBEGLIA et al. Static pickup and delivery problems: a classification scheme and survey. **TOP 15**, p. 1–31, 2007.

CHEN, J.; HU, X. Robotic process automation in modern warehouses. IEEE **Transactions on Automation Science and Engineering,** v.18(3), p. 897-909, 2021.

DANG, Q.V. et al. Scheduling a single mobile robot for part-feeding tasks of production lines. **Journal Intelligent Manufacturing,** v. 25, p. 1271–1287, 2014.

DANG, Q.V.; NGUYEN, C. T.; RUDOVÁ, H. Scheduling of mobile robots for transportation and manufacturing tasks. **Journal of Heuristics,** v. 25, p. 175-213, 2018.

DIANA, R. O. M.; SOUZA, S. R. Analysis of variable neighborhood descent as a local search operator for total weighted tardiness problem on unrelated parallel machines. **Computers & Operations Research,** v. 117, 2020.

ERDOĞAN, G.; BATTARRA, M.; LAPORTE, G. Scheduling twin robots on a line. **Naval Research Logistics**, n. 61, p. 119-130, 2014.

FERNANDES, H. A.; TAGLIALENHA, S. L. S. Sequenciamento de tarefas e agendamento de robôs móveis autônomos**.** *In*: ENEGEP 2021, ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO*,* v.1, **Anais ENEGEP 2021, Encontro nacional de engenharia de produção,** Foz do Iguaçu, 2021. Disponível em: https://www.abepro.org.br/biblioteca/TN_STO_356_1835_42671.pdf Acesso em: 3 jun. 2023.

FLEURY, M. T. L.; WERLANG, S. R. C.; LAPORTE, G. Pesquisa aplicada: conceitos e abordagens. **ANUÁRIO DE PESQUISA GVPESQUISA 2016-2017**, Abertura, 2017. Disponível em: https://periodicos.fgv.br/apgvpesquisa/article/view/72796 Acesso em: 17 nov. 2024.

GAO J.; SUN L.; GEN M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, **Computers & Operations Research**, v. 35, ed. 9, p. 2892-2907, 2008.

GIL, A. C. Como elaborar projetos de pesquisa, **Editora Atlas**, 4. ed. São Paulo, 2002.

GUROBI. **Gurobi Optimizer**. Disponível em: https://www.gurobi.com/products/gurobi-optimizer/. Acesso em nov. 2024.

HILLIER, F. S.; LIEBERMAN, G. J. **Introduction to Operations Research**. 9. ed. New York: McGraw-Hill. 2010.

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and applications, **European Journal of Operational Research**, v. 130, ed. 3, p. 449-467, 2001.

HANSEN, P.; MLADENOVIĆ, N. A Tutorial on Variable Neighborhood Search, **Groupe d'études et de recherche en analyse des décisions**, p. 1-26, 2003.

KAILER, D. L. O.; TAGLIALENHA, S. L. S. Heuristic method for the twin-robots scheduling problem in pickup and delivery systems. In: **Anais Do Simpósio Brasileiro De Pesquisa Operacional**, 2023, São José dos Campos. Anais eletrônicos... Campinas, Galoá, 2023. Disponível em: <https://proceedings.science/sbpo/sbpo-2023/trabalhos/heuristic-method-for-the-twin-robots-scheduling-problem-in-pickup-and-delivery-s?lang=pt-br> Acesso em: 14 Ago. 2024.

KANAGARAJ, G.; SHEIK MASTHAN, S.; YU, V. F. Meta-heuristics based inverse kinematics of robot manipulator's path tracking capability under joint limits. **Mendel**, v. 28, n. 1, p. 41-54, 2022.

LIU, M. et al. Multi-resource constrained flexible job shop scheduling problem with fixture-pallet combinatorial optimization, **Computers & Industrial Engineering,** v. 188, 2024

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search, **Computers & Operations Research**, v. 24, ed. 11, p. 1097-1100, 1997.

MLADENOVIĆ, N., TODOSIJEVIC, R., UROSEVIC, D. Less is more: basic variable neighborhood search for minimum differential dispersion problem. **Information Sciences,** v. 326, p. 160–171, 2016.

PATTNAIK, S. K.; MISHRA, D.; PANDA, S. A. Comparative study of meta-heuristics for local path planning of a mobile robot. **Engineering Optimization,** v. 54, n. 1, p. 134-152, 2022.

PINEDO, M. L. **Scheduling theory, algorithms, and systems**. 5. ed. New York: Springer Science. 2016.

SAARINEN, J., MONTONEN, P.; VALMET, A. Real-time control and monitoring of autonomous mobile robots in manufacturing. **Procedia Manufacturing**, v. 38, p. 49-56, 2019.

SAHU, B.; DAS, P. K.; KABAT, M. R. Multi-robot co-operation for stick carrying application using hybridization of meta-heuristic algorithm, **Mathematics and Computers in Simulation,** v. 195, p. 197-226, 2022.

SANTOS, E. J. **Análise de tempos de estabelecimento e ordem de manufatura no sequenciamento de tarefas em processos batelada**. 1994. Tese (Mestrado em 1994) – Faculdade de Engenharia Química, universidade estadual de campinas, Campinas, 1994.

SORENSEN, K.; SEVAUX, M.; GLOVER, F. A History of Metaheuristics, In: **Martí, R., Pardalos, P., Resende, M. (eds) Handbook of Heuristics. Springer, Cham.** 2018.

TAILLARD, É. D. **Design of Heuristic Algorithms for Hard Optimization**. Suíça: Springer. 2023.

TASGETIREN, M. F. et al. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. **European Journal of Operational Research,** v. 177, ed. 3, p. 1930-1947, 2007.

TERESHCHUK, V. et al. A scheduling method for multi-robot assembly of aircraft structures with soft task precedence constraints. **Robotics and Computer-Integrated Manufacturing,** v. 71, ed. 102154, 10 p., 2021.

THOMASSON, O. et al. Pallet location and job scheduling in a Twin-Robot system**. Computers & Operations Research**, v. 147, ed. 105956, 12 p., 2022.

THOMASSON, O. et al. Scheduling twin robots in a palletising problem**. International Journal of Production Research**, v. 56, n. 1-2, p. 518-542, 2018.

WANG, Z.; LIU, C.; GOMBOLAY, M. Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints. **Auton Robot**, v. 46, p. 249–268, 2022.

ZHAO, F.; ZHUANG, C.; WANG, L. e DONG, C. An Iterative Greedy Algorithm With Q-Learning Mechanism for the Multiobjective Distributed No-Idle Permutation Flowshop Scheduling, **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, vol. 54, n. 5, p. 3207-3219, 2024.

APÊNDICE A – Coeficiente Alpha

Algoritmo 4 Coeficiente Alpha.

Receber: i, i', k, k'
se $p_i \leq k$ **então**
  se $p_i \leq k$ **então**
    se $k < k'$ **então**
      $\alpha = 0$
    se não e se $p_i < k' \; e \; k' \leq k$ **então**
      $\alpha = (p_{i'} - k')\tau + 2\eta - (k' - p_i)\tau$
    se não e se $k' \leq p_i$ **então**
      $\alpha = (p_{i'} - k')\tau + 2\eta - (k' - pi)\tau$
  se não e se $p_{i'} < k'$ **então**
    se $k' \leq p_i$ **então**
      $\alpha = (p_i - k' + \sigma)\tau + (k' - p_{i'})\tau + 2\eta$
    se não e se $p_{i'} \leq p_i \; ou \; k' \leq k$ **então**
      $\alpha = (p_i - p_{i'} + \sigma)\tau + \eta$
    se não e se $pi' \leq k$ **então**
      $\alpha = (\sigma - (p_{i'} - p_i))\tau$
    se não
      $\alpha = 0$
se não e se $p_i < k$ **então**
  se $p_{i'} \leq k'$ **então**
    se $p_{i'} > p_i$ **então**
      $\alpha = 0$
    se não e se $p_{i'} \geq k'$ **então**
      $\alpha = (p_i - k' + \sigma)\tau + (k' - p_{i'})\tau + 2\eta$
    se não
      $\alpha = (p_i - p_{i'} + \sigma)\tau + \eta$
  se não e se $pi' \geq k$ **então**
    se $k' \leq p_i$ **então**
      $\alpha = (p_i - k' + \sigma)\tau + (p_{i'} - k')\tau + 2\eta$
    se não
      $\alpha = 0$
retorna: α

Fonte: Thomasson et al. (2018).

APÊNDICE B – Coeficiente Beta

Algoritmo 5 Coeficiente Beta.

1: Receber: i, i', k, k'
2: **se** $p_i \leq k$ **então**
3: **se** $p_{i'} > k'$ **então**
4: **se** $k' > k$ **então**
5: $\beta = 0$
6: **se não e se** $p_{i'} \leq k$ **então**
7: $\beta = 2\eta + \tau(k - p_i) + \tau(k - p_{i'} - \sigma)$
8: **se não**
9: $\beta = 2\eta + \tau(k - p_i) + \tau(p_{i'} - k) + \sigma - \eta$
10: **se não**
11: **se** $pi > k$ **então**
12: $\beta = 0$
13: **se não**
14: $\beta = 2\eta + \tau(k - p_i) + \tau(k - p_{i'} + \sigma)$
15: **se não se** $p_i > k$ **então**
16: **se** $p_{i'} > k'$ **então**
17: **se** $k' > p_i$ **então**
18: $\beta = 0$
19: **se não e se** $p_{i'} \leq k$ **então**
20: $\beta = 2\eta + \tau(p_i - k) + \tau(k - p_{i'} + \sigma)$
21: **se não se** $k' \leq k \;||\; p_{i'} \leq p_i$ **então**
22: $\beta = \tau(\sigma + p_i - p_{i'}) + \eta$
23: **se não**
24: $\beta = \tau(p_i - p_{i'} + \sigma)$
25: **se não**
26: **se** $p_{i'} > p_i$ **então**
27: $\beta = 0$
28: **se não se** $p_i \leq k$ **então**
29: $\beta = 2\eta + \tau(p_i - k) + \tau(k - p_{i'} + \sigma)$
30: **se não**
31: $\beta = \tau(\sigma + p_i - p_{i'}) + \eta$
32: **retorna:** β

Fonte: Thomasson et al. (2018).